TEKNILLINEN KORKEAKOULU

Sähkö- ja tietoliikennetekniikan osasto

Jani Kangas

SULAUTETUN OHJELMISTON SUUNNITTELUPROSESSIN KEHITYSHANKE

Diplomityö, joka on jätetty opinnäytteenä tarkastettavaksi diplomi-insinöörin tutkintoa varten Espoossa 24.10.2003

Työn valvoja

Professori Veikko Teikari

Työn ohjaaja

Diplomi-insinööri Panu Tuominen

Alkusanat

Tämä diplomityö on tehty Vaisala Oyj:n Instruments -divisioonan tuotekehitysosastolle. Työ tehtiin noin vuoden pituisesta organisaation kehityshankkeesta, joka alkoi helmikuussa vuonna 2003. Työn tarkoituksena oli tukea prosessiorganisaation kehittämistä.

Työn tekemiseen tarvitsin ja sain paljon tukea monilta tuotekehitysorganisaatioon kuuluneilta henkilöiltä. Haluankin nyt kiittää yhteisesti teitä kaikkia, jotka autoitte minua työn tekemisessä.

Erityinen kiitos kuuluu työni valvojalle professori Veikko Teikarille. Häntä haluan kiittää hänen asiantuntevista kommenteistaan sekä hänen tarjoamastaan mahdollisuudesta tehdä työ tästä kiinnostavasta aiheesta. Yhtä suuret kiitokset kuuluvat myös työni ohjaajalle Panu Tuomiselle, joka jaksoi lukea työni useaan otteeseen ja auttoi asiantuntevasti jopa sen pienienkin yksityiskohtien viilaamisessa.

Kiitokset saavat myös kehitysryhmän jäsenet Marko Hänninen ja Matti Kokki. He olivat erittäin tärkeitä tämän työn kannalta, koska ilman heitä ei kehityshanketta ja sitä kautta tätä työtä olisi voinut edes tehdä. Kiitoksen ansaitsevat myös ohjausryhmän jäsenet Jouni Rantanen, Heikki Joensuu ja Panu Kopsala, jotka ohjauksellaan pitivät projektin kasassa loppuun asti. Lisäksi kiitän Paul Wilkinsiä oikolukuavusta.

Suurimmat ja erityisimmät kiitokset saa tyttöystäväni Mari Laakso. Ilman hänen tukeaan, kannustustaan ja oikolukuapuaan tämä työ tuskin koskaan olisi valmistunut näin nopeasti. Haluan myös kiittää häntä tuesta, kuuntelemisesta sekä ymmärryksestä, jota hän on jaksanut osoittaa läpi koko opiskeluaikani. Haluan lisäksi kiittää koiraamme Kamua, joka aina uskollisesti häntää heiluttaen ja päätä kiinnostuneesti käännellen jaksoi kuunnella teoriaa nabloista prosessikonsultointiin.

Lopuksi haluan lausua lämpimimmät kiitokset perheelleni. Veljeäni Kaitsua haluan kiittää niistä kaikista neuvoista ja avusta mitä olen elämäni ja opiskelujeni aikana saanut. Vanhempiani Leenaa ja Einoa kiitän suuresti heidän vankkumattomasta tuesta sekä elämän opastuksesta, joka on loppujen lopuksi mahdollistanut tämän kaiken. Kiitos äiti ja isä!

Vantaalla, 24.10.2003 Jani Kangas

TEKNILLINEN KORKEAKOULU

DIPLOMITYÖN TIIVISTELMÄ

Tekijä:	Jani Kangas		
Työn nimi:	Sulautetun ohjelmiston suunnitteluprosessin kehityshanke		
Päivämäärä:	24.10.2003	Sivumäärä:	79
Osasto	Tuotantotalouden osasto		
Professuuri:	Johtaminen ja työpsykologia	Koodi:	Tu-53
Valvoja	Professori Veikko Teikari		
Ohjaaja	Diplomi-insinööri Panu Tuominen		

Diplomityön aiheena oli Vaisala Oyj:n Instruments -divisioonan tuotekehityksessä toteutettu sulautetun ohjelmiston suunnitteluprosessin kehityshanke. Työn tarkoituksena oli toimia prosessiorganisaation kehittämis- sekä ohjelmiston suunnitteluprosessimenetelmien valinnan ja käytön tukena.

Työn teoriaosuus painottuu kahteen pääalueeseen: organisaation kehittämiseen ja prosesseihin. Organisaation kehittämisen teoriaosuuden tarkoituksena on kriittisesti tarkastella kirjallisuutta apuna käyttäen organisaation kehittämisen menetelmiä ja niiden soveltuvuuksia käytännön kehitystyöhön. Ohjelmistoprosessiosuudessa puolestaan keskitytään kirjallisuudessa esitettyihin suunnitelmaohjautuva- ja Agile-metodeihin. Metodeja tarkastellaan erityisesti sulautetun ohjelmiston suunnittelun sekä Vaisala Instruments-tyyppisen organisaation kannalta.

Työn empiirisessä osuudessa kuvataan kehityshankkeen elinkaari alusta pilotointivaiheeseen saakka. Kuvauksessa pyritään esittämään tämän hankkeen kannalta kriittisimmät vaiheet ja ne menetelmät joilla sitä vietiin eteenpäin. Työn lopuksi analysoidaan kehityshankkeen sekä prosessin onnistumisen kriittisiä tekijöitä.

Tämän diplomityön tulos oli käytäntöön saattamista vaille valmis sulautetun ohjelmiston suunnitteluprosessi. Lisäksi työn tulosten pohjalta prosessi kuvattiin sekä dokumentointiin prosessimanuaaliksi, joka sisälsi mm. prosessikuvan, vaiheiden sanalliset kuvakset ja dokumenttipohjat.

Avainsanat: Sulautettu ohjelmisto, organisaation kehittäminen, prosessiorganisaatio, ohjelmiston suunnitteluprosessi, agile, prosessijohtaminen

HELSINKI UNIVERSITY OF TECHNOLOGY ABSTRACT OF THE MASTER'S THESIS

Author:	Jani Kangas		
Name of the thesis:	Development of an Embedded Software Design Process		
Date:	24.10.2003	Number of pages:	79
Department:	Department of Industrial Engineering and Management		
Professorship:	Work Psychology and Leadership	Code:	Tu-53
Supervisor:	Professor Veikko Teikari		
Instructor:	M. Sc. (Tech.) Panu Tuominen		

The subject of this Master's Thesis was the organization development (OD) project, which was made at Vaisala Oyj Instruments division. The OD-project's main focus was a software development process. The main objective of this thesis was to assist in the evaluation and usage of the organization and software development methods.

The presented theory is divided into two main sections: organization development and R&D-development processes. The organization development theories and their applicability to this project are reviewed in the OD related section. In the second theory section different plan-driven and Agile methods are reviewed. The main focus of the methods review was how they can be adapted to embedded systems programming, both in a general case, and in the specific case of Vaisala Instruments.

The progress of the development project is described in the empirical part of the thesis. It presents the methods used, and describes the phases of the project. The project's critical phases, and the selected software development process, are analyzed in detail at the end of the thesis.

The result of this thesis and OD-project is a viable software development process, which is ready for deployment. Additionally, this thesis was used as an information resource during the documentation of the software development process.

Keywords: embedded software, organization development, software development process, agile method, process management

SISÄLLYSLUETTELO

Alkusanat
Tiivistelmä
Abstract
Sisällysluettelo
Symbolit ja lyhenteet

1 JOHDANTO 1
1.1 Vaisala Oyj 2
1.2 Vaisala Instruments 2
1.3 Tutkimusongelma ja työn tavoitteet 3
1.4 Oma roolini kehityshankkeessa 3
1.5 Kehitysorganisaatio 3
1.5.1 Kehitysryhmä 3
1.5.2 Ohjausryhmä 3
1.6 Kehityshankkeen asiakkaat 3

2 MUUTOS JA KEHITTÄMINEN 5
2.1 Vaisalan sisäinen kehitysprosessi 5
2.2 Muutoksen ja kehittämisen aloittaminen 5
2.3 Nykytilan analysointi 6
2.4 Kehittäminen 7
2.5 Pilotointi 8
2.6 Muutosten toteuttaminen ja vakiinnuttaminen 8
2.7 Lopetus ja jatkuva parantaminen 9
2.8 Muutos- ja kehittämisteoriat vs. todellisuus 9
2.9 Organisaation kehitysmallien käyttö kehitystyön apuna 13

3 PROSESSIT JA YRITYSKULTTUURI 15
3.1 Prosessit 15
3.1.1 Projekti ja proseduuri vs. prosessi 15
3.1.2 Pää-, tuki ja johtoprosessit 16
3.2 Yrityskulttuuri 17

4 OHJELMISTON KEHITYSPROSESSIT 19
4.1 Ohjelmistoprosessikäsitteen määritelmä 19
4.2 Historia 20
4.3 Olio-ohjelmointi ja kehitysprosessit 22
4.4 Suunnitelmaohjautuvat lähestymistavat 22
4.4.1 Vesiputousmalli 22
4.4.2 Spiraalimalli 24
4.5 Agile-tyyppinen lähestymistapa 25
4.5.1 XP 25
4.5.2 Evo 27
4.5.3 Feature Driven Development 28
4.5.4 Adaptive Software Development 29
4.6 Muut ohjelmiston kehitysprosessit 30
4.7 Yhteenveto ohjelmistoprosesseista 31

5 OHJELMISTOSTANDARDIT JA LAATU 33
5.1 ISO 9001 33
5.2 ISO 9000-3 33
5.3 ISO/IEC 12207 34
5.4 ISO/IEC TR 15271 34
5.5 IEEE 829 34
5.6 Ohjelmistojen turvallisuusstandardit 34
5.7 Laatujohtaminen 35

6 TUOTEKEHITYSPROSESSIEN JOHTAMINEN 36
6.1 Prosessijohtamisen määritelmä 36
6.1.1 Toimintajärjestelmä 37
6.1.2 Osaaminen 38
6.1.3 Ihmissuhteet 38
6.2 Ohjelmiston suunnitteluprosessin johtaminen 38
6.3 Prosessimittarit 40
6.4 Prosessi vallan välineenä 41

7 KEHITYKSEN KULKU 42
7.1 Kehityshankkeen aloitus 42
7.1.1 Kehitysstrategia (Road Map) 42
7.1.2 Kehityskatselmus 0 (DVR0) 42
7.2 Analyysivaihe (Analysis) 43
7.2.1 Nykytilan arvioinnin suunnittelu 44
7.2.2 Workshopin sisältö 45
7.2.3 Workshopin tulokset 46
7.2.4 Workshopin tulosten yhteenveto 46
7.2.5 Nykytilan hyvät ja huonot puolet 48
7.2.6 Benchmarking 49
7.2.7 Prosessimittarien kehittäminen 51
7.2.8 Kehityskatselmus 1 (DVR1) 52
7.3 Kehitysvaihe 54
7.3.1 Suunnitteluprosessin kuvaaminen 55
7.3.2 Suunnitteluprosessin sanallinen kuvaus 56
7.3.3 Dokumenttipohjien suunnittelu 59
7.3.4 Organisaation muutos 59
7.3.5 Kommentointikierrokset 60
7.3.6 Pilottiprojektien suunnittelu 62
7.3.7 Kehityskatselmus 2 (DVR2) 62
7.4 Pilotointivaihe (Pilot Projects) 64
7.4.1 Pilotointivaiheen tulokset 64
7.4.2 Kehityskatselmus 3 (DVR3) 65

8 YHTEENVETO JA JOHTOPÄÄTÖKSET **66**
8.1 Kehityshankkeen kulku ja onnistuminen 66
8.2 Ohjelmiston suunnitteluprosessin sisältö 71

9 LOPUKSI **75**
9.1 Mietteitä 75
9.2 Jatkotoimenpiteet 77
9.3 Kohti jatkuvaa parantamista 78
9.4 Tulevaisuus 79

LÄHTEET **80**

LIITTEET **87**
Liite 1, Workshopin aikataulu ja sisältö 87
Liite 2, Henkilökohtaiset kysymykset 91
Liite 3, Ryhmän 1 tekemä nykytilan prosessikuvaus 93
Liite 4, Ryhmän 2 tekemä nykytilan prosessikuvaus 94
Liite 5, Ensimmäinen kehitysryhmän yhteinen luonnos suunnitteluprosessista 95
Liite 6, FAQ-kysymykset 96

Symbolit ja lyhenteet

ASD	Adaptive Software Development
BR	Business Review
CM	Change Management
CMMI	Capability Maturity Model Integration
DR	Design Review
DVR	Development Review
Evo	Evolutionary Project Management methods
EIMS	Expertice, Interaction, Management ja Self
FDD	Feature Driven Development
Metodi	Systemaattinen tapa tehdä jotakin (Hidding, 1998)
OD	Organization Development
Ohjelmiston elinkaari	Ohjelmiston elinkaari kestää ohjelmiston asiakasvaatimusten alkukohdasta ohjelmistotuotteen ylläpidon lopettamiseen saakka
OMT	Object Modeling Technique
RUP	The Rational Unified Process
SEI	Software Engineering Institute
SKP	Sulautetun ohjelmiston Kehitys Projekti
SPI	Software Process Improvements
SPICE	Software Process Improvement and Capability dEtermination
Sulautettu järjestelmä	Ohjelmistosysteemi joka on "upotettuna" laitteeseen, jota ohjelmisto ohjaa
XP	Extreme Programming

1 JOHDANTO

"Ongelmanratkaisu- ja keksimistyölle ominaista on se, että yhtä oikeaa tai parasta ratkaisua ei välttämättä ole olemassa - on vain joukko toteutettavissa olevia, eri tavoin yhtä päteviä ratkaisuja. Työssä on hyväksyttävä sen epävarmuus ja epäselkeys."
Miia Martinsuo, TkT, 2001.

Tämän diplomityön tavoitteena oli edistää sekä toimia tiedonlähteenä Vaisala Instruments -divisioonan tuotekehitysosastolla tehdyssä sulautetun ohjelmiston suunnitteluprosessin kehityshankkeessa. Lisäksi työn tarkoituksena oli dokumentoida tehty kehitystyö ja näin auttaa tulevaisuudenkin kehityshankkeita.

Kehityshankkeen päämääränä oli kehittää tuoteprosessin aliprosessi, joka kuvaa ja ohjaa tulevaisuudessa Instruments-divisioonan tuotteisiin tulevien sulautetun ohjelmistojen suunnittelun kulkua. Työ lähti liikkeelle vuoden 2003 alussa tilanteessa, jossa sulautetun ohjelmiston suunnittelulla ei ollut olemassa kuvattua prosessia.

Yllä oleva Martinsuon artikkelista otettu lainaus kuvaa mielestäni erittäin hyvin tätä työtä ja sen ongelmakenttää. Työtä tehdessä tuli nimittäin selväksi, että ei ole olemassa yhtä ja oikeaa tapaa suunnitella sulautettua ohjelmistoa. Suunnittelumetodeja ja -tyylejä oli tarjolla runsaasti ja jokaisella niistä tuntui olevan sekä hyvät että huonot puolensa. Ominaista niille oli myös se, että jokaiselle tavalle löytyi vannoutuneita kannattajia, jotka omilla tai muiden esittämillä esimerkeillä pyrkivät todistamaan kyseisen tavan parhaaksi ja oikeaksi vaihtoehdoksi. Yksi työn tarkoituksista olikin selvittää, mikä näistä tavoista tai niiden yhdistelmästä olisi paras tapa Vaisala Instrumentsin tapauksessa.

Tällä työllä on kahden tyyppisiä lukijoita. Toiset lukevat työtä organisaation kehittämisen kannalta, kun taas toiset lukevat työtä ohjelmistoprosessien näkökulmasta. Tässä työssä ei tehdä selkeätä linjanvetoa näkökulmien välillä, vaan työssä lähestytään ongelmakenttää pitäen mielessä kumpikin näkökulma. Tämän työn perimmäiseksi tarkoitukseksi voikin määritellä ohjelmistoprosessiajatteluun perustuvien teorioiden käytäntöön saattamisen organisaation kehittämisen teorioiden avulla. Työn toivottiin näin tuovan syvempää näkemystä perinteiseen organisaation kehittämiseen sekä samalla tarjota ohjelmistoprosesseista enemmän kiinnostuneille tapoja niiden toteuttamiseksi.

Työn teoriaosuus on luonnollisesti myös jakautunut kahteen erityyppiseen aihealueeseen eli organisaation kehittämiseen ja prosesseihin. Työn teoriaosuudessa tullaan esittelemään muutoksen ja kehittämisen malleja sekä analysoimaan niiden käytettävyyttä organisaation kehittämisessä. Teoriaosuudessa myös tutustutaan ja analysoidaan ohjelmiston suunnitteluprosesseja, standardeja sekä prosessijohtamista. Tämän tarkoituksena on antaa kehitystyölle taustaa ja selventää kehityshankkeen aikana tehtyjen ratkaisujen taustalla vaikuttaneita tekijöitä. Teoriaosuuden yhteydessä on myös analysoitu esiteltyjä teorioita Vaisala Instruments -divisioonan kannalta. Näin on pyritty löytämään tapoja teorioiden tehokkaampaan hyödyntämiseen kehityksessä.

Työn empiirisessä osiossa esitellään teoriaosuuteen perustuen tehdyt toimenpiteet sekä niistä saatuja tuloksia. Empiirinen osio pyrkii myös kuvaamaan todellisia kehityksen aikaisia tapahtumia sekä kriittisiä vaiheita.

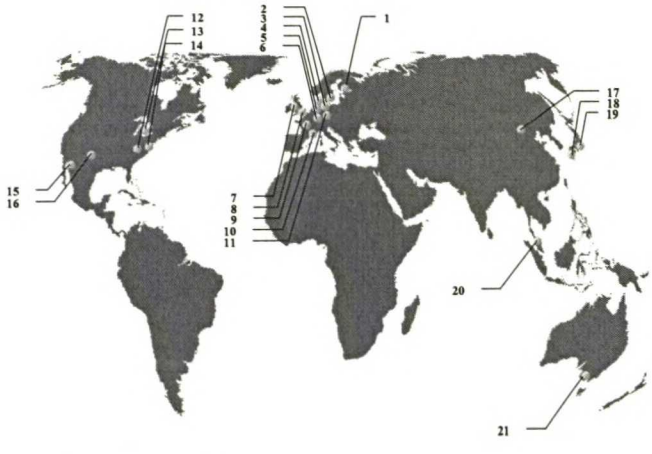
Työn empiirinen osuus päättyy saatujen tulosten analysointiin. Tarkoituksena on eritellä työn todelliset saavutukset sekä mahdolliset puutteet ja tarkastella saatujen tuloksien mielekkyyttä ja todenmukaisuutta.

Lopuksi työssä pohditaan tulevaisuuden näkymiä, auki jääneitä kysymyksiä ja esitetään jatkotoimenpide-ehdotukset. Jatkotoimenpide-ehdotuksien tarkoituksena on auttaa kehityshanketta sen jalkauttamisvaiheessa.

1.1 Vaisala Oyj

Vaisala Oyj on vuonna 1936 perustettu kansainvälinen teknologiakonserni, joka kehittää ja valmistaa elektronisia mittausjärjestelmiä ja -laitteita. Tuotteiden sovellusalueina ovat meteorologia, ympäristötieteet, liikenne ja teollisuus. Vaisalan sisällä organisaatio on jakautunut neljään eri divisioonaan Vaisala Soundings, Vaisala Solutions, Vaisala Remote Sensing ja Vaisala Instruments.

Vaisalassa työskentelee noin 1200 henkilöä. Toimipisteitä sillä on yhteensä 21 ja ne sijaitsevat 11 eri maassa (Kuva 1). Vaisalan liikevaihto vuonna 2002 oli 196,2 Miljoonaa euroa. Viennin osuus liikevaihdosta oli 96,3%.



Kuva 1 Vaisalan toimipisteet maailmalla.

1.2 Vaisala Instruments

Vaisalan Instruments kehittää ja valmistaa ympäristöä mittaaviin antureihin perustuvia mittalaitteita. Instrumentsin tuotekehitystoimintoja sijaitsee Suomessa sekä Saksassa.

Tunnetuimmat ympäristösuureet, joita laitteilla mitataan, ovat lämpötila, kosteus, kastepiste, paine, hiilidioksidi ja tuuli. Vaisalan Instrumentsin tekemät mittalaitteet ovat tarkoitettu enimmäkseen teollisuuden käyttöön.

1.3 Tutkimusongelma ja työn tavoitteet

Tämän työn tavoite on määritellä ja kuvata Vaisala Instrumentsin sulautetun ohjelmiston suunnitteluprosessin kehityshakkeen aikaiset työvaiheet. Lisäksi tavoitteena on dokumentoida ohjelmiston suunnitteluprosessi sekä toimenpiteet sen jalkauttamiseksi. Työ pyrkii myös selventämään kehityksen kulkua ja tekijöitä, jotka vaikuttivat saatuun lopputulokseen.

1.4 Oma roolini kehityshankkeessa

Kehitystyöryhmä koostu neljästä henkilöstä ja sitä ohjaa kolmen hengen ohjausryhmä. Omaa rooliani tässä kehitystyöryhmässä kuvaisi ehkä parhaiten ns. muutosagentin rooli (Huczynski & Buchanan, 2001; Kosonen et al, 1998). Tämä siksi, että tehtäväni on auttaa ja ohjata kehitysryhmää viemään muutosta läpi: tukemalla sitä organisaation kehittämis- ja ohjelmistoprosessiosaamisella, edistämällä organisaation sisäistä kommunikointia, tuomalla esiin muiden kehitysryhmäläisten osaamisen sekä tekemällä tarvittaessa kaikkia kehityshankkeeseen liittyviä työtehtäviä. Mainittakoon vielä, että roolinikin takia tämän diplomityön lopullinen konkreettinen tulos ei ole oma saavutukseni, vaan se on koko hankkeeseen osallistuneiden yhteinen tulos.

1.5 Kehitysorganisaatio

Hankeen läpiviemiseksi määriteltiin kehitysorganisaatio. Tällä tarkoitetaan tässä työssä kehitys- sekä ohjausryhmää.

1.5.1 Kehitysryhmä

Kehitystyöryhmä koostui neljästä henkilöstä. Panu Tuominen toimi tämän kehitysryhmän vetäjänä. Kaikki muut henkilöt allekirjoittanut mukaan lukien ovat samalta osastolta olevia ohjelmiston suunnittelijoita.

1.5.2 Ohjausryhmä

Kehitystyöryhmän toimintaa ohjasi ja valvoi ohjausryhmä. Ohjausryhmään kuului kolme henkilöä, jotka olivat tuoteprosessin omistaja, ohjelmistointressiryhmän vastuuhenkilö sekä yksi projektipäällikkö.

1.6 Kehityshankkeen asiakkaat

Kehityshankkeelle voidaan eritellä kolme asiakasta. Hankkeen tilaaja sekä ensimmäinen kontaktiasiakas (contact clients) on ohjausryhmä ja ehkä tarkemmin Vaisala Instrumentsin tuotekehityksen prosessin omistaja. Pääasiakkaita (primary clients) ovat Instrumentsin tuotekehityksen sulautetun ohjelmiston suunnittelijat, koska kehityshanke tulee vaikuttamaan tähän noin kymmenen hengen suuruiseen joukkoon varsin suoraan esimerkiksi työtapojen ja toivottavasti myös asenteiden muutoksilla. Hankkeen lopullisena asiakkaana (ultimate clients) voidaan pitää koko Vaisalan or-

ganisaatiota tai Instruments -divisioonaa. Tämä siksi, että hanke tulee toivottavasti vaikuttamaan positiivisesti koko organisaation ja divisioonan tulokseen sekä laatuun pitkällä tähtäimellä. (Schein, 1987.)

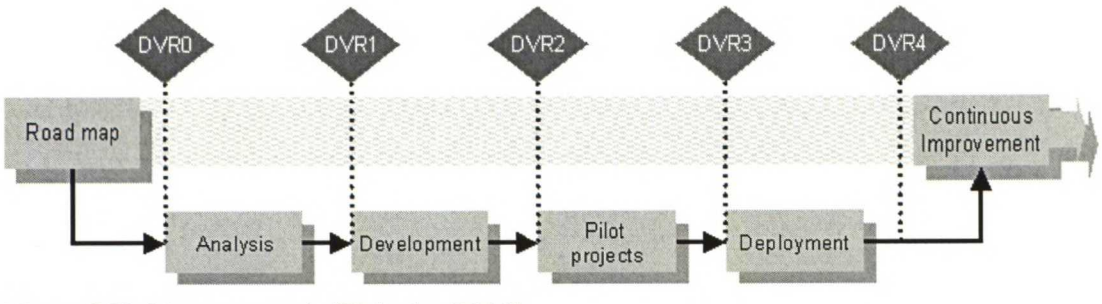
2 MUUTOS JA KEHITTÄMINEN

Tämä diplomityö on yksi raportti kehityshankkeesta, joka on viety läpi tavoitteellisella sekä vaiheittaisella muutoksen ja kehittämisen menetelmällä (Lanning et al, 1999; Lanning, 2001). Tämä teoriaosuuden luku esittelee organisaation kehittämisen teorioiden taustoja sekä viitekehyksen niistä toimintatavoista, joita käytettiin apuna työn lopputulokseen pääsemiseksi.

Luvun alussa esitellään lyhyesti hankkeessa käytetty kehitysprosessi. Käytetty kehitysprosessi ja sen työvaiheet tulevat esiin yksityiskohtaisemmin työn empiirisessä osiossa, joten tämän luvun pääpaino on enemmän tavoitteellisen kehittämisen kirjallisuuden menetelmien läpikäymisessä ja tulkitsemisessa. Tämä luku päätetään esitettyjen teorioiden todenmukaisuuden, käyttökelpoisuuden ja taustalla vaikuttavien tekijöiden analysointiin.

2.1 Vaisalan sisäinen kehitysprosessi

Kuvassa 2 on esitelty Vaisalan sisäisessä käytössä oleva prosessien kehitysprosessi. Kehittäminen alkaa visioinnilla ja strategisella suunnittelulla (road map). Kun tarvittavat kehityskohteet ovat löytyneet ja ne on ajoitettu tiekarttaan (road map), aloitetaan varsinainen kehittäminen. Kehittäminen aloitetaan analyysilla (analysis), jonka tarkoituksena on löytää todelliset kehityksen kohteet sekä rajata kehitysosa-alueet. Analyysivaihetta seuraa kehitysvaihe (development), jossa nimensä mukaan kehitetään analyysivaiheessa diagnosoidut kehityskohteet. Kun kehitysvaihe on saatu päätökseen, siirrytään pilotointivaiheeseen. Pilotoinnin tarkoituksena on katsoa, kuinka kehitetty prosessi toimii käytännössä. Pilotoinnin avulla voidaan vielä myös tarvittaessa viimeistellä ja muuttaa prosessia. Viimeisenä vaiheena on kehitetyn prosessin käytäntöönotto (deployment). Tämän jälkeen kehitysprojekti päätetään ja aloitetaan jatkuva kehitys (countinuous improvement). Jatkuvalla kehityksellä tarkoitetaan uusien kehitysprojektien aloittamista tai esimerkiksi parannuksien tekoa vanhoihin prosesseihin. (Vaisala, 2000.)



Kuva 2 Kehitysprosessi. (Vaisala, 2000)

2.2 Muutoksen ja kehittämisen aloittaminen

Muutoksen aloittaa muutostarve. Muutostarve voi olla esimerkiksi tyytymättömyys toimintaprosesseihin sekä prosessien tuotoksiin tai vaikkapa organisaation ulkopuo-

lelta tuleva uhka. Muutostarpeen tulkitseminen voi olla vaikeaa ja ehkä vielä vaikeampaa voi olla saada ihmiset ymmärtämään muutoksen tarpeellisuus. Muutoksen tarpeen sisäistämistä voi esimerkiksi tehokkaasti estää ihmisten luonnollinen pelokkuus muutosta kohtaan tai muut sen tuomat epävarmuustekijät eli muutosvastarinnat. (Beer et al, 1990; Buchanan & Badham, 1999; Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999; Schein, 1987.)

Muutostarpeen tullessa ilmi sitä aletaan analysoida. Analysoinnin pohjalta ruvetaan rakentamaan visiota. Tarpeeksi laaja-alaisen ja sitä kautta hyvän vision aikaansaamiseksi analysoinnissa ja vision rakentamisessa voidaan ja on syytä käyttää hyväksi johdon, henkilöstön tai esimerkiksi ulkopuolisten konsulttien osaamista. (Denton, 1996; Kosonen et al, 1998; Kotter, 1995.) Tässä on kuitenkin syytä olla varuillaan, sillä kuten Lanning et al. toteavat, esimerkiksi vuosien kokemus yrityksen toiminnasta ei välttämättä anna oikeaa tietoa kehityskohteista vaan saattaa sitä vastoin estää todellisten ongelmien näkemisen (Lanning et al, 1999).

Vision rakentamisessa ja uskottavuuden luomisessa on myös tärkeää yrityksen merkittävien henkilöiden tuen saaminen uudelle visiolle. Lanning et al. lisäksi korostavat, että onnistunut kehitysprojekti lähtee aina yrityskohtaisesta muutostarpeesta, eikä esimerkiksi päivän muoti-ilmiöistä. (Beer et al, 1990; Denton, 1996; Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999.)

Hyvä ja oikein toteutettu visio antaa muutostarpeelle oikean suunnan sekä konkretisoi sitä tarvittavan yksityiskohtaisesti. Lisäksi on muistettava, että vision tarkoituksena ei ole siirtää ulkopuolista painetta tai uhkaa organisaatioon vaan sitä vastoin kehittää keinoja tulevaisuuden menestymisen takaamiseksi. Vision avulla on perustellusti pystyttävä muuttamaan muutostarpeet mielekkäiksi ja konkreettiset tavoitteet omaaviksi kehitysprojekteiksi. Voidaan esimerkiksi tehdä tiekartta (kts. Road Map Kuva 2), jossa suunnitellut kehityshankkeet ovat ajallisesti sijoitettu loogiseksi sekä toisiaan tukevaksi kokonaisuudeksi. (Kosonen et al, 1998; Kotter, 1995.)

2.3 Nykytilan analysointi

Visioon pyrkiminen kannattaa aloittaa vasta, kun kehitysprojektin konkreettiset tavoitteet ovat selvillä ja johdon tuki projektille on varmistettu. Jos nämä asiat ovat kunnossa laaditaan strategia eli kehityssuunnitelma tavoitteeseen pääsemiseksi. Kehityssuunnitelman pitäisi sisältää mm. projektin tavoitteet, rajaus, aikatalutus, kehittämismenetelmät, organisaatio ja käytettävät resurssit. (French & Bell, 1999; Kosonen et al, 1998; Lanning et al, 1999; Kotter, 1995.)

Kehitysprojekti aloitetaan yleensä nykytilan analysoinnilla. Tämän tavoitteena on selvittää mitä yrityksessä todella tehdään ja mitkä ovat sen todelliset kehitystarpeet. Nykytilan analyysin apuna voidaan käyttää erilaisia menetelmiä, joita ovat mm. teemahaastattelut, ryhmätyöt, olemassa oleviin dokumentteihin tutustumiset, kvantitatiiviset menetelmät tai epäviralliset keskustelut. Analysoinnissa on pyrittävä analysoimaan oikeita asioita sekä samalla on vältettävä kehitysryhmän tai kehittäjän omien näkemysten ja tulkintojen sekoittumista kehitettävän kohteen ongelman tulkintaan. Analysointia voidaan tehdä joko yrityksen omin resurssein tai siinä voidaan

käyttää apuna ulkopuolisia konsultteja. (French & Bell, 1999; Kosonen et al, 1998; Schein, 1987.)

Analyysista saatu tieto on tärkeää tulevaisuudessa, kun arvioidaan kehityshankkeen onnistumista. Sen avulla voidaan osoittaa tarvittaessa, että kehitysprojektista oli todellista hyötyä ja sillä voidaan myös arvioida projektin kehittymistä. Yksi analysoinnin ja projektin seuraamisen ongelma on toisaalta se, että kaikkea ei voi mitata numeroilla. Esimerkiksi monet ympäristöarvot, prosessin hallittavuus, yhteistyön todellinen toimivuus tai hyvinvointi ovat vaikeasti mitattavissa. (Kosonen et al, 1998; Lanning et al, 1999.)

Nykytilan analyysin tuloksesi pitäisi saada selvitys siitä, mitkä ovat tulevan kehitysvaiheen kehitysosa-alueet, jotta kehittämisessä ei tuhlattaisi aikaa ja resursseja vääriin asioihin. Yhtälailla tärkeää on analyysin perusteella osata rajata pois ne osa-alueet, mihin kehitysprojektilla ei ainakaan tulla ottamaan kantaa. Analyysivaiheen tulisi myös selkeyttää sekä vahvistaa yrityksen näkemystä visiosta. (Beer et al, 1990; Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999; Vaisala, 2000.)

2.4 Kehittäminen

Muutostarpeen ongelman ratkaisuun eli kehittämiseen ja suunnitteluun voidaan ryhtyä, kun konkreettinen visio on sisäistetty ja nykytila analysoitu. Mikään ei tietystikään estä tekemästä esittämiäni vaiheita toistensa lomassa, mutta kehittämisvaihe on toisaalta järkevä aloittaa vasta edellisten vaiheiden jälkeen.

Kehittäminen ja uusien menetelmien suunnittelumenetelmät ovat erittäin riippuvaisia kehityskohteesta ja siitä mitä halutaan kehittää. Ei voida siis esittää yhtä ja parasta tapaa suunnitella asioita. Organisaation kehittämisen teoria kuitenkin tarjoaa tähänkin vaiheeseen työkaluja ja menetelmiä tehdä tehokasta suunnittelua. Tästä ehkä jo klassikoksi muodostunut esimerkki on Scheinin kehittämä prosessikonsulttimainen kehitystyön ohjaaminen ja parantaminen (Schein, 1987). Lisäksi on olemassa lukuisia muita menetelmiä kuten esimerkiksi Change Management (CM) tyyli, joka painottaa perinteisen OD-metodien lisäksi myös ulkopuolisen substanssiosaamisen ja monitaitoisten kehitystiimien hyödyntämistä kehittämisessä (Worren et al, 1999).

Suunnitteluvaiheen aikana korostuu kehittäjän sekä ryhmän sisäiset roolit. Kehitystyö asettaa erilaisia osaamisvaatimuksia kuin henkilöstön tavallinen työ, mikä on syytä ottaa huomioon kehitystyöryhmää koottaessa. Tarvittaessa on syytä panostaa ryhmän koulutukseen tai ottaa mukaan henkilö tai konsultti joka omaa organisaation kehittämisen taitoja, jotta kehittämisestä saataisiin tehokasta. Vaihtoehtoisesti isoille sekä vakinaisille kehitystyöryhmille voidaan käyttää esimerkiksi foorumeita tai muita kokouksia siten, että niihin kutsutaan aina sillä hetkellä kehitettävän osa-alueen asianomaisia tai asiantuntijoita. (Kosonen et al, 1998; Schein, 1987.)

Kehitystyöryhmän jokapäiväinen toiminta perustuu periaatteessa tavalliseen projektijohtamiseen ja -hallintaan. Esimerkiksi tässä työssä noudatettava kehitysprosessikaavio (Kuva 2) havainnollistaa kehitysprojektia ja sen ohjausryhmän toimintaa. Tässä työssä ei tulla teorian kannalta tämän enempää perehtymään organisaation kehittämisryhmien toimintaan tai siihen kuinka niiden toimintaa voidaan parantaa esi-

merkiksi prosessikonsulttimaisesti tai CM-tyylisesti toimimalla. (Kosonen et al, 1998; Lanning et al, 1999; Schein, 1987; Vaisala, 2000; Worren et al, 1999.)

Kehitysvaiheen lopputulokseksi pitäisi saada dokumentoitu sekä selkeä ratkaisu organisaatiolle tehtävistä muutoksista ja toimintatavoista sen pilotoinniksi sekä vakiinnuttamiseksi. Pelkästään dokumentoitu ja ohjeistettu ratkaisu ei kuitenkaan riitä. Kehitysvaiheessa joudutaankin tekemään paljon töitä esimerkiksi ihmisten sitouttamiseksi muutokseen. Näitä toimia ovat esimerkiksi henkilöstön osallistaminen, psykologisen turvan luominen sekä muut muutokseen motivoivat keinot (Schein, 1987; Buchanan & Badham, 1999). Lisäksi kehitysvaiheen aikana projektin etenemisestä tiedottaminen on tärkeässä roolissa ihmisten mukaan saamisessa. Denton esimerkiksi artikkelissaan toteaa, että hyvällä kehityksen aikaisella tiedottamisella estetään ikävien huhujen leviäminen, jotka voivat pahimmassa tapauksessa torperoida hanketta. (Denton, 1996). Mutta, jos kehittäminen on hoidettu mallikkaasti tai ainakin sitä yrittäen, voidaan aloittaa kehitetyn uuden toimintatavan jalkauttaminen. (Beer et al, 1990; Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999; Vaisala, 2000.)

2.5 Pilotointi

Pilotoinnin tarkoituksena on kokeilla käytännössä suunniteltua toimintatapaa. Pilottiprojekteja käytetään, koska näin voidaan minimoida riskejä sekä tarvittaessa tehdä vielä muutoksia sekä korjauksia lopulliseen, muun organisaation kattavaan toimintatapaan. (Kosonen et al, 1998; Lanning et al, 1999; Vaisala, 2000.)

Vaarana pilottivaiheessa on ns. kultapossukerho-ilmiö, jossa muu organisaatio alkaa hylkiä pilottiprojektia, koska se saa erityskohtelua. Vaarana voi myös olla, että esimerkiksi samaan aikaan ylläpidettävä entinen toimintatapa voi aiheuttaa lisätyötä ja tätä kautta voi uusi tapa joutua huonoon valoon. Toisaalta pitää muistaa, että onnistunut pilottiprojekti voi tarjota näyttöä onnistuneesta kehittämisestä, mikä voi helpottaa hankkeen läpiviemistä muualla organisaatiossa. (Kosonen et al, 1998; Lanning et al, 1999.)

2.6 Muutosten toteuttaminen ja vakiinnuttaminen

Muutosten toteuttaminen ja vakiinnuttaminen on kehityshankkeen ehkä vaativin vaihe ja siinä punnitaan kuinka hyvin edellisissä vaiheissa on taustatyötä tehty. Vaiheen vaativuutta sekä raskautta kuvaa myös hyvin monesti organisaation kehittämisen luennoilta tuttu sanonta: "Hyvin suunniteltu on vain korkeintaan puoliksi tehty!". Kehittämisprojektien ongelmaksi voikin muodostua, että panostetaan kivoihin vaiheisiin kuten visiointiin ja suunnitteluun, jolloin jalkauttamisvaiheelle jää vain vähän motivaatiota, aikaa ja resursseja. Toteutuksen ja vakiinnuttamisen tekeekin vaikeaksi se, että siihen ei ole olemassa suoria työkaluja tai menetelmiä. Jalkauttamisen onnistuminen tai epäonnistuminen onkin kiinni paljolti kehittäjien henkilökohtaisista kyvyistä ja haluista saada aikaan konkreettisia tuloksia. (Kosonen et al, 1998.)

Toteutusvaihe alkaa yleensä ennen kuin kehittämisvaihe on kerinnyt kokonaan loppua. Muutoksia ja vakiinnuttamista toteuttaessa on syytä varautua pieniin sekä isoihin ongelmiin, koska esimerkiksi toimintatapoja voidaan joutua muuttamaan tai hienosäätämään aivan loppuvaiheissakin. Tämän takia onkin hyvä, jos toimintatapoja

pystytään vähän kerrassaan etukäteen jalkauttamaan, koska tällöin jää enemmän aikaa menetelmien hiomiseen ja viilaamiseen. (Kosonen et al, 1998; Lanning et al, 1999.)

Kun muutosta lähdetään vakiinnuttamaan, on johdon merkitys siinä tärkeä. Johdon on loppuun asti sitouduttava muutokseen sekä omalla toiminnallaan pyrittävä edistämään muutosta sekä lisäksi kehitettävä muutosta tukevia toimintoja kuten esimerkiksi palkkio ja mittausjärjestelmillä. Johdon toiminnan lisäksi esimerkiksi fyysisillä, järjestelmä- ja organisaatiomuutoksilla on tärkeä rooli muutosten vakiinnuttamisessa. (Denton, 1996; French & Bell, 1999; Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999.)

Sekä Kosonen et al., Kotter että Lanning et al. painottavat nopeiden tulosten saamisen tärkeyttä. On siis pystyttävä mahdollisimman pian kehityshankkeen jalkauttamisen jälkeen osoittamaan uuden toimintatavan hyödyt. Näin saadaan johto vakuuttuneeksi hankkeen hyödyllisyydestä ja mikä tärkeintä tekijät uskomaan uusien tapojen hyödyllisyyteen. (Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999.)

2.7 Lopetus ja jatkuva parantaminen

Vaikka tavoitteena on jatkuva kehittyminen, on yksittäiset kehitysprojektit syytä päättää muodollisesti. Projektin päättämisellä ja loppuraportoinnilla on erityisesti symbolinen merkitys, jotta ei pääse syntymää ns. ikuisia kehitysprojekteja. Loppuraportissa tulisi kertoa mitä projektin aikana tuli tehtyä sekä mitä siitä on tulevaisuudessa opittavaa. On myös syytä tehdä loppuraporteista tarvittaessa lyhennelmiä tai lyhyitä esitelmiä, jotka kertovat projektin tuotokset ja saavutukset pähkinänkuoressa sekä selkeästi. Kosonen et al. vielä muistuttavat, että loppuraportista vetää jokainen aina tulkintoja oman näkökulman kannalta, joten riippuen näkökulmasta voidaan kehitystyön tulos nähdä negatiivisena tai positiivisena. Tämä on myös syytä tiedostaa, koska edellisten kehityshankkeiden mielekkyys ja tulokset vaikuttavat aina tulevien hankkeiden käynnistämiseen ja ihmisten motivoitumiseen kehitystyöhön. (Kosonen et al, 1998; Lanning et al, 1999; Vaisala, 2000.)

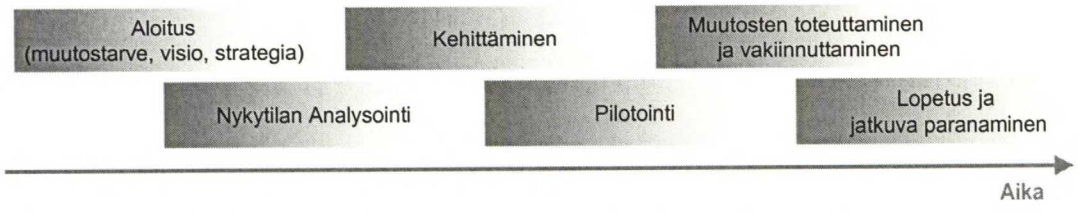
Koska projekti on aina ainutlaatuinen tapahtuma, on raportti kirjoitettava sellaiseksi, että sitä voidaan hyödyntää ja siitä voivat muut organisaatiossa olevat ottaa opiksi. Lisäksi loppuraportointi antaa tärkeää palautetta kehitystyöryhmälle sen tekemästä työstä ja sitä kautta se kehittää heitä kehittäjänä. (Kosonen et al, 1998; Lanning et al, 1999.)

Loppuraportin jälkeen voidaan siirtyä jatkuvan parantamisen vaiheeseen tai aloittaa uusia kehitysprojekteja. Päättämisen ja loppuraportin avulla työntekijä voi siirtyä eteenpäin työtehtävissään, koska hän tuntee saaneensa jotain päätökseen. Samalla hän ehkä saa enemmän otetta joskus erittäin abstraktistakin kehitystyöstä.

2.8 Muutos- ja kehittämisteoriat vs. todellisuus

Tässä luvussa on muutoksen läpiviemiseksi esitelty järjestelmällinen ja kuuteen eri vaiheeseen vaiheistettu malli (Kuva 3). Vaiheet on sijoitettu loogiseen järjestykseen ja ne hiukan lomittuvat, mutta seuraavaan tehtävään siirtymiseksi on lähes aina vaa-

dittu kuitenkin edellisen vaiheen täydellistä loppuun suorittamista. Onko tämä malli sitten ideaalinen, kun kerran se voidaan löytää monista alan kirjoista? Onko olemassa muita tapoja ajatella kehityksen kulkua ja eroavatko ne toisistaan? Missä vaiheessa todellinen muutos syntyy ja miten? Mitä pitäisi ottaa huomioon kehitysmalleja noudatettaessa? Onko edes olemassa oikeaa tai ideaalista tapaa kuvata muutosta?



Kuva 3 Vaiheittaisen muutoksen vaiheiden lomittainen sijoittuminen ajan suhteen.

Edellä on joukko kysymyksiä, joihin ei varmasti voida antaa kattavaa vastausta tämän työn puitteissa. Niihin pyritään kuitenkin vastaamaan auttavasti ja todistamaan, miksi esittämäni ja Vaisalan malli voivat hyvinkin toimia käytännössä, vaikka ne eivät kuvaakaan kaikkea organisaation kehittämisen työvaiheita yksityiskohtaisesti tai täydellisesti.

Lähdetään liikkeelle tarkastelemalla, miten kehittämistä ja muutosta voidaan katsella eri näkökulmista. Van de Ven ja Poole ovat jaotelleet organisaation kehitysmalliteoriat ja menetelmät niiden muutoksen moottorin avulla neljään eri luokkaan. Ne ovat elämänkaari- (life-cycle), teleologiset- (teleological), dialektiset - (dialectical) ja evoluutioteoriat (evolutionary). (Van de Ven & Poole, 1995.)

Tässä luvussa läpikäydyt teoriat ovat lähinnä teleologista lähestymistapaa. Sille tunnusomaista on vision luominen ja visioon pyrkiminen lähes keinolla millä hyvänsä. Teleologisille malleille onkin ominaista esittää kriittisiä polkuja, eli toimenpiteitä jotka tulee ainakin suorittaa haluttuun lopputulokseen pääsemiseksi. Esimerkiksi Kotter ja Beer et al. ovat esittäneet malleja jotka kertovat lähinnä mitkä ongelmat pitää ratakaista visioon pääsemiseksi. Ne jättävätkin kehittäjän vastuulle vaiheiden pituuksien ja lomittamisen hallinnan ja määrittämisen. (Beer et al, 1990; Kotter, 1995.)

Vaisalassa käytettävä kehitysprosessi voisi puolestaan olla orjallisesti noudatettuna suhteellisen lähellä elämänkaarityyppistä mallia, koska elämänkaarimallille on ominaista se, että kaikki vaiheet tulee suorittaa loppuun ennen kuin voidaan siirtyä seuraavaan vaiheeseen. Tätä käyttäytymistä vielä voimistaa DVR-tilaisuudet. Van de Ven ja Poole vielä toteavat, että jos aiotaan yhdistellä tai käyttää samassa projektissa hyväksi eri teorioita on syytä olla tarkkana, että käyttää samaa muutoksen moottorin omaavia kehitysmalleja. Muussa tapauksessa yhdistämisestä voi olla jopa haittaa tai se ei tuo ainakaan uudelle syntyneelle mallille lisäarvoa. (Van de Ven & Poole, 1995.)

Alan kirjallisuudessa esitetyt teoriat voitiin jakaa neljään eri tyyppiin muutoksen moottorien mukaan, mutta lisäksi alan kirjallisuus voidaan jaotella kahteen luokkaan, voidaan nimittäin puhua akateemisesta ja konsulttikirjallisuudesta. Akateemiselle

kirjallisuudelle on ominaista kehityksen kompleksisointi, kyseenalaistaminen ja sen toteuttamisen vaikeuden painottaminen, kun taas konsulttikirjallisuus pyrkii yksinkertaistamaan ja puolustelemaan teorioitaan onnistuneilla hankkeillaan (success stories). Akateemista kirjallisuutta lukevat alan tutkijat, kun taas konsulttikirjallisuus on suunnattu alan konsulteille, johtajille ja yrityksen kehittäjille. Mainittakoon, että tässä työssä esitetty muutoksen ja kehityksen mallit sekä myöhemmin esiteltävät ohjelmistoprosessimallit ovat taustoiltaan lähempänä konsulttikirjallisuuden tuotoksia kuin akateemista. Tämän takia työtä kirjoittaessani olenkin pyrkinyt harjoittamaan kovaakin lähdekritiikkiä konsulttikirjallisuutta lukiessani ja lainatessani. (Miller & Greenwood, 1997.)

Loppujen lopuksi ei voida sanoa, kumpi kirjallisuus on oikeassa. Kummassakin suunnassa on varmasti omat etunsa ja haittansa. Konsulttikirjallisuuden edustajat joutuvat myymään kirjojaan ja oikeasti toteuttamaan kehityshankkeita ja tätä kautta heillä saattaakin olla erittäin syvällinen näkemys alasta. Toisaalta heidän elantonsa on kiinni kirjojen ja konsulttikäyntien määrästä, joten kertovatko he aina koko totuutta, jos se vaikuttaisi kirjan myyntiin? Akateemisten tutkijoiden puolestaan on helppo arvostella konsultteja tai epäonnistuneita hankkeita, koska he eivät yleensä saa palkkaansa tutkittavasta kohteesta, joten heillä ei ole ns. oma lehmä ojassa. Toisaalta akateeminen lähestymistapa saattaa johtaa turhan syvälliseen ja kriittiseen tutkimukseen, josta ei sitten käytännön kehittämistyössä ole välttämättä hyötyä. Voidaan myös karkeasti todeta, että akateeminen kirjallisuus analysoi mitä on tehty kun taas konsulttikirjallisuus pyrkii saamaan asiat vain tehdyksi. (Lanning, 2001; Miller & Greenwood, 1997.)

Miller ja Greenwood toteavat myös artikkelissaan, että konsultti ja akateemiset kirjallisuudet käsittävät muutoksien vaikutuksen muuhun organisaatioon eri tavalla. Konsulttikirjallisuudelle on ominaista pitää organisaatiota löyhänä kokonaisuutena, jonka osia voidaan muuttaa ilman että muut osat häiriintyvät. Akateeminen kirjallisuus puolestaan painottaa, että muuttamalla yhtä kohtaa aiheutetaan väistämättä muutoksia muuallekin organisaatiota. Tämä ehkä myös selittää osaltaan sen miksi akateemista kirjallisuutta pidetään kompleksisempana. Miller ja Greenwood lisäksi jatkavat, että muutos on kyllä helppo saavuttaa esimerkiksi käyttäen konsulttikirjallisuutta, mutta saatu tulos ei välttämättä ole hyödyllinen. Lisäksi jokainen muutos lisää epävarmuutta ja riskejä kuten Amburgey et al. ovat artikkelissaan tutkineet (Amburgey et al, 1993). (Miller & Greenwood, 1997.)

Olkoon kehittäjä sisäinen, akateeminen tai konsultti ja noudattakoon hän sitten akateemista tai konsulttikirjallisuutta tai mitä muutoksen moottoria hyvänsä, on hänen tai heidän lopullinen tehtävänä viedä muutos läpi organisaatiossa. Muutoksen ja sen etenemisen hahmottamiseen on monia eri tapoja. Inns on artikkelissaan jakanut hahmottamistavat kahden tyyppisiksi matkametaforiksi. Hänen mukaansa on päämäärätietoisia matkoja ja "tutkimusretkeilijämatkoja" eli prosessiorientoituneisuutta. Hyvä esimerkki tästä on Lanning et al. kirja jonka otsikkona lukee: "Matkaopas muutokseen" (Lanning et al, 1999). Päämäärätietoiset matkametaforateoriat tarjoavat selkeän reitin kuinka yritys saavuttaa päämäärän kulkemalla niiden viitoittamaa tietä. Prosessiorientoituneet teoriat puolestaan painottavat sitä, että on pyrittävä tekemään asiat vain hyvin sekä pienissä lyhyissä pyrähdyksissä päämäärä aina mielessä, jolloin lopulta matka päättyy johonkin "tuntemattomaan", joka on toivottavasti haluttu sekä organisaation kannalta paras. Prosessiorientoituneet menetelmät ovat siis lähellä jat-

kuvan kehityksen ajattelutapaa, kun taas matkametafora lähempänä radikaalia tai episodimaista kehitystä. Inns vielä mainitsee, että päämäärätietoinen lähestymistapa on konsulttikirjallisuudelle ominaisempaa. Tätä samaa päätelmää Miller ja Greenwood perustelivat sillä, että konsulttikirjallisuus luo johtajille tunteen, että valta on heidän käsissään ja luo kehittämiselle myyttiä sankarillisesta muutosjohtamisesta. (Inns, 1996; Miller & Greenwood, 1997; Weick & Quinn, 1999.)

Edellä on pohdittu teorioiden tapaa ja todenmukaisuutta esittää muutosta ja kehittämistä. Mutta onko se organisaatiolle luonnollista? Hannah ja Freeman ovat nimittäin esittäneet, että organisaatiolle on tyypillisempää pysähtyä, koska niiden sisäisen inertia pyrkii aina stabiloimaan tilanteen. Inertian voimakkuus sekä vaikutus muutoksiin riippuu organisaation koosta, kompleksisuudesta ja ympäristöstä. (Hannan & Freeman, 1984.) Tämän tyyppinen ajattelu johtaa siihen, että organisaatiossa ei tapahdu kehitystä kuin silloin, kun muutoshankkeita on käynnissä. Weick ja Quinn kyseenalaistavatkin tämän käsityksen muutoksesta, eli siirtymisen stabiilista tilasta muutoksen kautta takaisin stabiiliin tilaan. Heidän mukaansa nykypäiväiseen kompleksiseen maailmaan sopii paremmin jatkuvan muutoksen malli, joka huomioi sen, että taustalla oleva organisaatio muuttuu koko ajan vaikka sille ei tehtäisikään mitään. Toisin sanoen tästä voisi päätellä, että nykypäivän organisaatioissa muutosta pitäisi tehdä ohjaamalla organisaatiota prosessiorientoituneesti. (Inns, 1996; Weick & Quinn, 1999.)

Episodimaisia muutosmallien, joista ehkä tunnetuin on Kurt Lewinin 1950 luvulla esittämä sulata - muuta - jäädytä -malli, hyödyllisyyttä ja todellisia onnistumismahdollisuuksia on kritisoitu. Dunphy ja Stace artikkelissaan väittävät, että onnistunut episodimainen muutos vaatii lähes aina vähintään jonkin asteisen kriisin, koska muuten organisaatiossa ei pystytä saamaan aikaan tarpeeksi motivaatiota tai valmiutta muutokseen. Voisi myös sanoa, että kriisiä tarvitaan juuri inertian voittamiseen. Dunphy ja Stace myös puolustivat prosessiorientoitunutta kehitystä varsinkin, jos organisaatiossa tarvitaan vain pieniä korjauksia ja aikaa on riittävästi. Toisaalta heidän esittämäänsä mallia on kritisoitu ympäristön liialla yksinkertaistamisella ja johtajien roolin muutokseen vähättelyllä (Dunford et al, 1990). (Dunphy & Stace, 1988; Hannan & Freeman, 1984; Inns, 1996; Weick & Quinn, 1999.)

Miten ja millä muutos sitten voidaan saada aikaan? Olkoon kehittämiskäytäntö tai kehitetty uusi toimintatapa miten hyvä tahansa, ei sillä ole tulevaisuutta, jos todelliset vallankäyttäjät siihen usko tai sitä puolusta. Kehittäjän on omassa toiminnassaan ja loppuratkaisun suunnittelussa huomioitava sekä tunnistettava, ketkä todellista valtaa käyttävät. Suunnittelemalla esimerkiksi sellaisen prosessin, joka talloo jonkun sosiaalisesti vaikutusvaltaisen organisaation jäsenen varpaille voi johtaa tilanteeseen, jossa uusi prosessi ei koskaan tule toimimaan organisaatiossa halutulla tavalla. Kehitysvaiheessa hyvä kehittäjä käyttääkin valtaa ja kehitysmenetelmiä yhdessä kehityshankkeen läpiviemisen helpottamiseksi. Valta ja sitä kautta poliittiset pelit ovat siten oleellinen osa kehittämistä, joten niitä pitää osata käyttää ja hyödyntää oikein. (Buchanan & Badham, 1999.)

Pohditaan vielä lopuksi, milloin muutos todella tapahtuu. Käytännössä helpoin tapa ajatella on se, että muutos tapahtuu, silloin kun uusi toimintatapa on julkistettu, koulutettu ja ihmiset alkavat sitä muodollisesti noudattamaan. Näin yksinkertaisesti voitaisiin ajatella, mutta Barret et al. ovat artikkelissaan pohtineet tätä ajatusta syvälli-

semmin. Heidän mukaan muutos tapahtuu sosiaalisen konstruktion kautta. Toisin sanoen muutos tapahtuu vähitellen, kun ihmiset alkavat saamaan vanhoille tai uusille käsitteille uusia merkityksiä ja sisältöä. Tämän työn osalta tämä voisi tarkoittaa sitä, että esimerkiksi kehitystyöryhmä löytää uuden merkityksen sanoille suunnitelmaohjautuva ja agile, sekä mitä ne tarkoittavat ohjelmistoprosesseista ja heidän omasta suunnittelutyöstä puhuttaessa. Kehittäjän näkökulmasta tämä tarkoittaa puolestaan sitä, että minun on autettava ryhmää käsitteiden määrittämisessä sekä niiden merkityksen luomisessa juuri Vaisalan ja tämän kehityshankkeen kannalta ajateltuna. (Barret et al, 1995.)

2.9 Organisaation kehitysmallien käyttö kehitystyön apuna

Kappaleessa 2.8 on kriittisesti analysoitu organisaation kehitysprosesseja ja niiden ominaisuuksia ja toimivuutta. Kappaleen lähteenä toimineet artikkelit olivat pääosin akateemista kirjallisuutta ja kritiikin kohteena olivat yksinkertaistetut muutoksen mallit ja lähinnä konsulttikirjallisuus. Kappaleen tarkoituksena olikin tuoda kritiikkiä sekä ajatuksia insinöörimäiseen lähestymistapaan ja vaiheittaiseen kehitykseen kuten esimerkiksi Vaisalan kehitysprosessiin (Vaisala, 2000).

Vaisalan malli on varmasti käyttökelpoinen ja sitä tukevat monet kirjallisuuden teoriat. Vaiheittainen malli kuitenkin tunnetusti yksinkertaistaa kehittämistä ja tekee siitä lähinnä evoluutiomaisen (Lanning, 2001; Van De Ven & Poole, 1995). Miller ja Greenwood antavatkin artikkelissaan viisi ohjetta (Taulukko 1), joita konsulttikirjallisuuden yksinkertaistettuja malleja käyttävän kehittäjän on syytä pitää mielessä.

Taulukko 1 Realistisemman vaiheittaisen muutoksen lisävaiheet/huomioitavat asiat. Mukailtu: (Miller & Greenwood, 1997)

1. Transformaalinen muutos on paljon vaativampaa, kuin mitä yleensä väitetään.
2. Poliittiset pelit (valta) on syytä huomioida muutosta läpi vietäessä.
3. Muutoksen taloudelliset hyödyt on syytä kartoittaa etukäteen.
4. On syytä pohtia myös nykyisen tilan kannattavuutta, ennen muutoshankkeeseen ryhtymistä.
5. On syytä pitää mielessä, että ei ole yhtä ja oikeaa tapaa tehdä muutosta.

Huolestuttavinta mielestäni Vaisalan mallissa on sen yksinkertaisuus ja lineaarisuus, joka tuo insinöörille virheellisen turvallisuuden tunteen, jolloin hän ei osaa varautua organisaation kehittämisen todellisiin ongelmiin ja haasteisiin. Malli ei mielestäni myöskään selkeästi painota jalkauttamisen tärkeyttä ja sitä, että työ ei lopu siihen kun uusi prosessi on julkistettu.

Vaisalan malli on kuitenkin mielestäni erittäin hyvä ja selkeä pohja muutoksen toteuttamiseen ja voi vain kuvitella mitä tuloksia saataisiin, jos mallina olisi esimerkik-

si "tutkimusmatkailumetafora" tai jokin muu "vapaampi malli". Mielestäni malli esittää hyvin kehityksen kriittisen polun ja osaa myös hyödyntää projektimaista lähestymistapaa katselmuksien sekä ohjausryhmän toimintojen avulla. Mallia käyttävien kehittäjien on kuitenkin syytä pitää mielessä, että asiantuntijatyötä helpottamaan tehtyjen prosessikuvausten ei ole tarkoitus kuvata ja dokumentoida kaikkea prosessin aikana ja taustalla tapahtuvaa (Martinsuo, 2001; Laamanen, 2001). Näihin seikkoihin perustuen uskallan sanoa, että Vaisalan mallia voidaan käyttää tehokkaasti prosessien kehitykseen kunhan kehittäjät, kehitys- ja ohjausryhmä omaavat hyvät kehittämis- sekä substanssiosaamistaidot ja pitävät mielessä taulukossa 1 esitetty asiat. (Lanning et al, 1999; Lanning, 2001.)

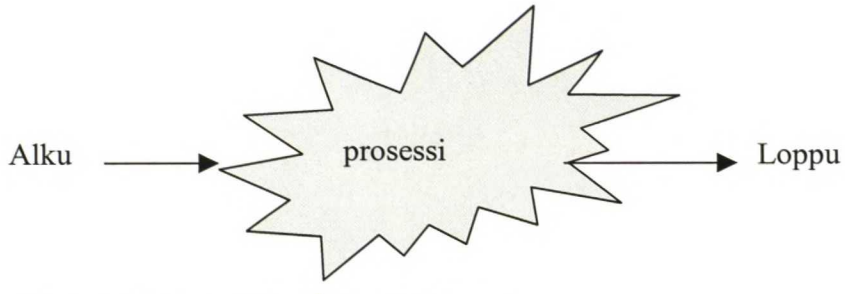
3 PROSESSIT JA YRITYSKULTTUURI

Perehdytään seuraavassa prosessi- ja yrityskulttuurikäsitteisiin. Näiden käsitteiden määrittelyjä ei voida luonnollisestikaan ohittaa, kun puhutaan prosessiorganisaation kehittämisestä. Kulttuurikäsite on myös tämän työn kannalta erittäin olennainen, koska ohjelmointitapoja ei oltu tätä työtä ennemmin kuvattu tai mallinnettu. Tämän takia voitaisiin sanoa, että ohjelmointitavat olivat Vaisala Instrumentsin kulttuuriin sidonnaisia (Schein, 2001).

3.1 Prosessit

Käsitettä prosessi käytetään kirjallisuudessa erittäin paljon ja se usein sekoitetaankin tahallisesti tai tahattomasti käsitteisiin projekti, tuotanto, yms. Seuraavassa esitetty prosessi käsitys perustuu suurimmaksi osaksi Wil van der Aalstin ja Kees van Heen ja Robert G. Cooperin esittämiin määritelmiin.

Prosessi on lyhyesti sanottuna kokoelma työtehtäviä, tiloja, aliprosesseja ja niiden keskinäisiä vaikutuksia. Prosessilla on aina oltava yksi alku ja yksi loppu kuten kuvassa 4 on esitetty. (Aalstin & Heen, 2002.)



Kuva 4 Prosessissa on yksi alku ja yksi loppu.

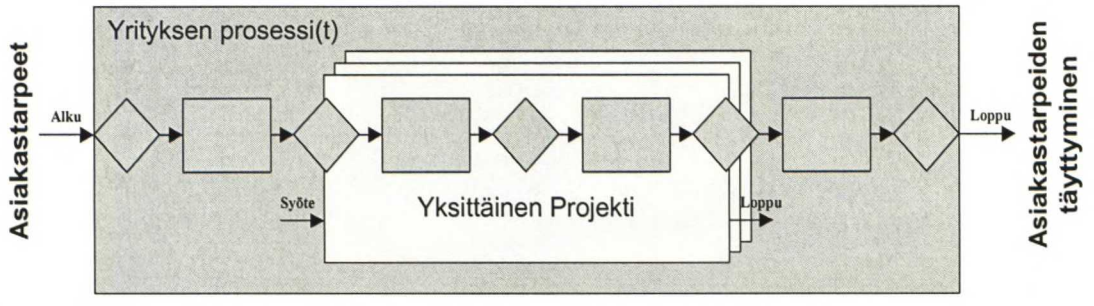
Prosessin sisällä olevat komponentit muodostavat yhdessä loogisen kokonaisuuden. Jokainen prosessin komponentin vuorovaikutus vaikuttaa aina omalla toiminnallaan prosessin lopputulokseen. Prosessi itsessään ei voi olla eristetty ympäristöstään vaan se on aina upotettuna ympäristöönsä, jonka kanssa se on vuorovaikutuksessa. (Smart et al, 1996.)

3.1.1 Projekti ja proseduuri vs. prosessi

Aalstin ja Heen toteavat, että prosessia voidaan kutsua myös proseduuriksi. Heidän mielestään prosessilla ja proseduurilla ei ole suurta eroa, koska kummatkin käsitteet määrittävät edellä esitetyn prosessin määritelmän. (Aalstin & Heen, 2002.) Tässä työssä kuitenkin käytetään sanaa prosessi juuri edellä esitetyn määritelmään viitatessa.

Projektin ja prosessin ero on yleensä mielletty selkeämmäksi. Miia Martinsuo erotteli projektin ja prosessin eron seuraavasti. "Prosessi on vaiheet ja päätökset tuotteen tai

palvelun kehittämiseksi (alusta - loppuun). Projekti puolestaan on ainutkertainen, sisältö-, tavoite- (aika, kustannus, laatu) ja resurssipuittein rajattu prosessin toteuma." Kuva 5 havainnollistaa projektin ja prosessin eroa. (Martinsuo, 2003.)



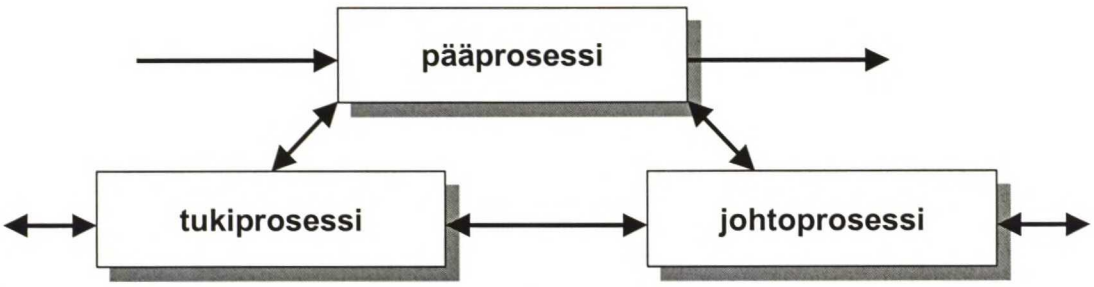
Kuva 5 Prosessin ja projektin ero. Mukailtu: (Martinsuo, 2003)

3.1.2 Pää-, tuki ja johtoprosessit

Prosessit voidaan erotella pää-, tuki ja johtoprosesseihin. Pääprosessit (Primary processes) ovat niitä, jotka tuottavat yrityksen tuotteen tai palvelun. Nämä prosessit tuovat tuottoa yritykselle ja ovat yleensä asiakasorientoituneita. Esimerkkejä pääprosesseista on esimerkiksi suunnittelu, tuotteen tai palvelun myynti tai tuotanto. (Aalstin & Heen, 2002.)

Toissijaiset prosessit eli tukiprosessit (Support processes) sanan mukaisesti tukevat pääprosessia. Tukiprosesseja voivat olla esimerkiksi koneen tai tuotannon ylläpitoprosessit. Vastaavasti tukiprosesseja voi olla esimerkiksi rekrytointi, koulutus tai rahoitushallinto. (Aalstin & Heen, 2002.)

Johtoprosessit (Managerial processes) ohjaavat ja koordinoivat pää- sekä tukiprosesseja. Tämän ohella johtoprosessien tarkoituksena on määrittää tavoitteet sekä edellytykset muille prosesseille. Johtoprosessi pitää myös sisällään rahoitus- sekä muut hallinnolliset kontaktit. Kuva 6 havainnollistaa eri prosessien välisiä suhteita. (Aalstin & Heen, 2002.) Laamanen kirjassaan tosin kritisoi johtoprosessiajatusta. Tämä siksi, että prosessin tärkein tehtävä olisi palvella asiakkaita, eikä pönkittää tai alleviivata jotain prosessia ja henkilöitä joita siihen kuuluu. Johtoprosessi myös hämärtää käsitystä, jonka mukaan prosessien pitäisi parantaa itseohjautuvuutta. (Laamanen, 2001.)

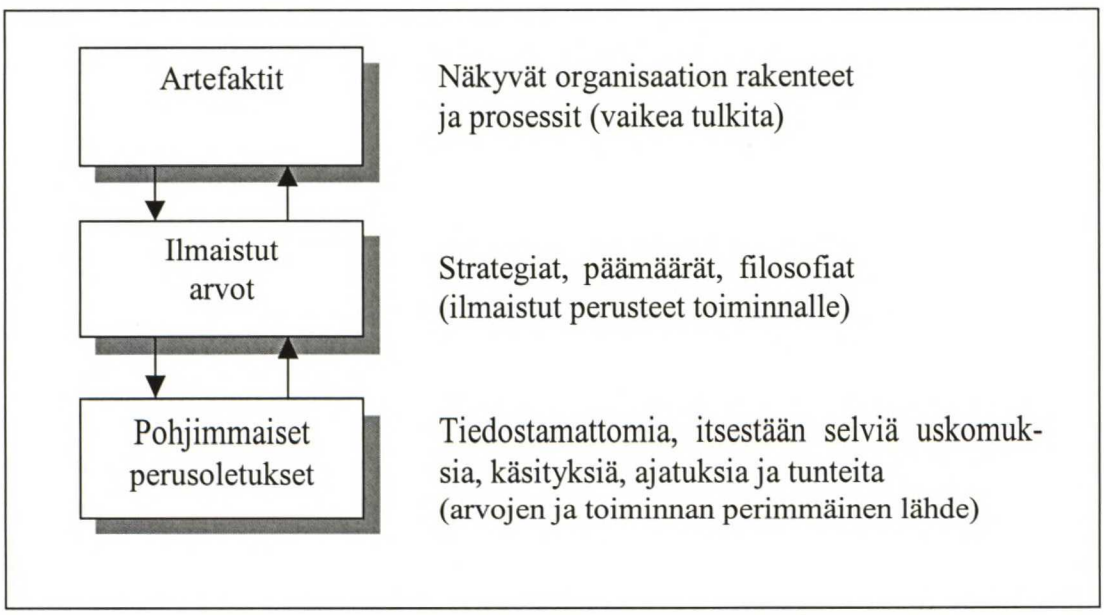


Kuva 6 Kolmen erityyppisen prosessin yhteys. (Aalstin & Heen, 2002)

3.2 Yrityskulttuuri

Nykyisestä Vaisala Instrumentsin ohjelmiston suunnittelu käytännöstä ei ole olemassa mitään dokumentoitua prosessia. Voitaisiin sanoa, että nykyisin kaikki suunnittelu perustuu divisioonan sisäiseen ohjelmistosuunnittelukulttuuriin Vaisalassa (Schein, 2001).

Kuva 7 esittää Scheinin kehittämää kulttuurimallia. Nykyisestä tuotekehityksen tilanteesta voidaan tunnistaa artefaktit sekä ilmaistut arvot ja joskus myös näitä ohjaavat pohjimmaiset perusolettamukset. (Schein, 2001.) Artefakteina voidaan pitää tuoteprosessia, jonka mukaan nykyisin ohjelmistoa suunnitellaan. Ilmaistuna arvona voidaan pitää sitä, että jokainen suunnittelija sanoo noudattavansa tuoteprosessia ja tekevänsä myös laadukasta työtä. Pohjimmaiset perusolettamukset puolestaan tarkoittavat sitä mitä oikeasti tehdään, eli ei noudateta tuoteprosessia tai luistetaan dokumentoinnista ja testauksesta.



Kuva 7 Kulttuurien tasot. (Schein, 2001)

Tunnettu prosessien kehittäjä Robert G. Cooper on kirjassaan todennut, että prosessin muutos on yksi vaikeimmista organisaation muutoksista, koska sillä on suora vaikutus kulttuuriin. (Cooper, 1999.) Myös Extreme Programming (XP) kehittäjä Kent Beck toteaa kirjassaan, että suurin este XP:n tehokkaaseen käyttöönottoon on kulttuuri (Beck, 2000). Voitaisiin myös sanoa, että koko yrityksen toiminta ja tehokkuus perustuu loppujen lopuksi yrityksessä vaikuttavaan tai vaikuttaviin kulttuureihin (Yukl, 1998). Näiden seikkojen takia kulttuurin vaikutusta tai olemassaoloa ei voi kiistää tai aliarvioida tehtäessä prosessien kehitystä (Jones, 2002).

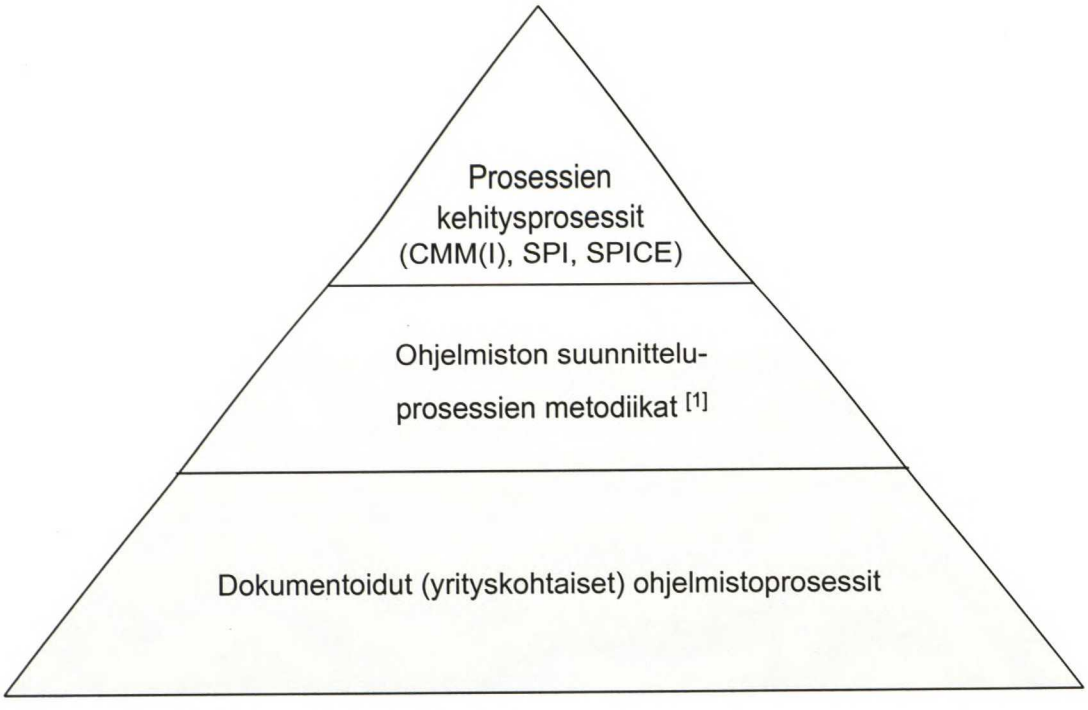
4 OHJELMISTON KEHITYSPROSESSIT

Seuraavaksi käsitellään tämän työn kannalta tärkeimpiä ohjelmiston kehitysprosesseja sekä niiden ominaisuuksia. Aloitetaan käsittely määrittelemällä mitä tarkoitetaan ohjelmistokehitysprosessilla ja tämän jälkeen kerrotaan lyhyesti ohjelmistoprosessien historiasta. Lopuksi esitellään metodiikat, jotka on eritelty karkeasti kahteen eri luokkaan eli suunnitteluohjautuviin ja agile-tyyppisiin metodiikkoihin (Boehm, 2002).

Metodeista[1] kerrotaan niiden perusominaisuudet, elämänkaarimalli sekä niiden soveltuvuus sulautetun ohjelmiston suunnitteluun. Lopuksi tehdään yhteenveto sekä pyritään selvittämään mitkä metodit ja mallit soveltuisivat parhaiten kehityshankkeemme ongelmaan.

4.1 Ohjelmistoprosessikäsitteen määritelmä

Tämän kehityshankkeen aikana tuli useaan otteeseen ongelmia ohjelmistoprosessimääritelmän kanssa. Tätä varten on syytä ensin selkeyttää mitä tarkoitetaan, kun puhutaan ohjelmistoprosesseista ja kirjallisuudessa esiintyvistä lyhenteistä ja menetelmistä. Kuvassa 8 on pyritty havainnollistamaan eri käsitteiden eroja ja yhteyksiä.



Kuva 8 Ohjelmistoprosessikäsitepyramidi.

Pyramidin kuvassa 8 ylimmäksi olen laittanut ohjelmiston kehitysprosessien kehitysprosessit. Näitä ovat esimerkiksi CMMI ja SPICE. CMMI on ns. jatkuvan parantamisen malli. Sen avulla yritys pystyy määrittämään, missä "kypsyystilassa" yrityksen ohjelmistoprosessi on ja mitä yrityksen tulee tehdä, jotta se saavuttaisi seuraavan kehittyneemmän tilan. Näitä tiloja on CMMI:ssä määritelty viisi kappaletta. Lyhyesti

[1] metodi = systemaattinen tapa tehdä jotakin (Hidding, 1998)

sanottuna yrityksen prosessi on CMMI:n ensimmäisessä tilassa (initial) silloin, jos sen prosessia ei ole sen kummemmin määritelty tai kuvattu. Lopulta, kun yritys on päässyt viidenteen tilaan (optimizing) tarvitsee ohjelmistoprosessia mallin mukaan enää vain optimoida ja virittää.[1] SPICE on, samoin kuin CMMI, jatkuvan parantamisen malli. SPICE perustuu ISO 15504-standardiin. Siinä pyritään myös määrittelemään ohjelmistoprosessin kypsyys ja sitä kautta löytämään prosessin parannuskohteita.[2] On myös syytä mainita, että SPICE ja CMMI eivät ota kantaa käytettävään suunnitteluprosessiin.

CMMI ja SPICE ovat erittäin raskaita ja byrokraattisia kehittämisen malleja ja soveltuvat parhaiten lähinnä puhtaasti ohjelmistoa tuottavan yrityksen kehittämiseen. Niitä ei tulla käsittelemään tässä työssä tämän enempää.

SPI (Software Process Improvement) on myös mainittu pyramidiin huipulla. Lyhenne esiintyy ohjelmistokirjallisuudessa erittäin paljon ja sillä tunnutaan nykyisin viitamaan kaikkeen toimintaa, jotka tähtäävät ohjelmistoprosessien kehittämiseen.

Pyramidin keskivaiheille olen sijoittanut ohjelmiston suunnitteluprosessien taustalla olevat metodiikat. Näillä tarkoitan esimerkiksi vesiputous- tai agile-tyyppisiä ajatusmalleja sekä menetelmiä.

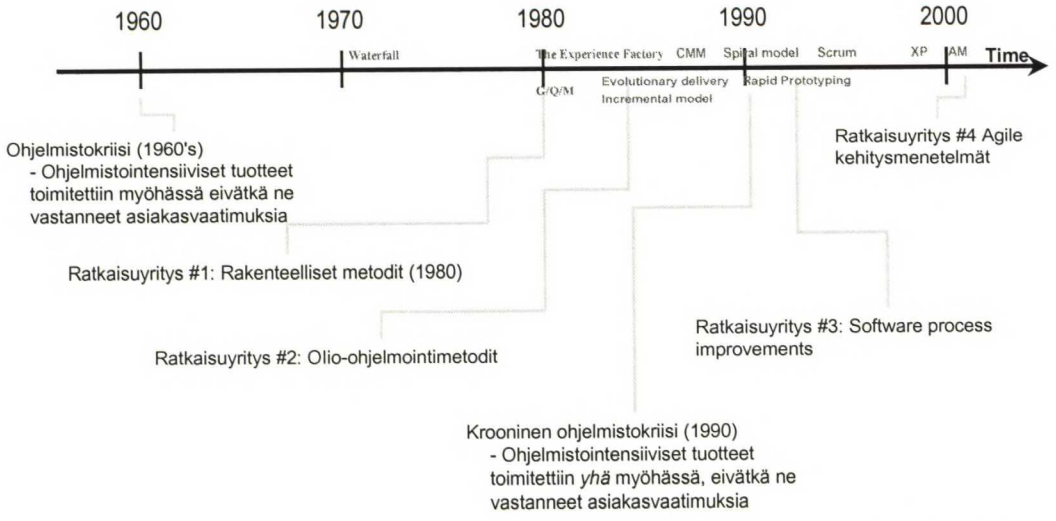
Pyramidin alimmaksi olen sijoittanut yritysten käyttämät ohjelmiston suunnitteluprosessit. Ne ovat yrityksen omaan tuotekehitykseen räätälöityjä prosesseja, jotka yleensä pohjautuvat johonkin ohjelmistokehitysmetodiikkaan tai -menetelmään. Suunnitteluprosesseja pyritään kehittämään muuttamalla taustalla olevaa metodiikkaa sekä käyttämällä apuna esimerkiksi eri SPI, CMMI tai SPICE prosessin kehitysmenetelmiä. Ohjelmistoteollisuuden prosessikehityksessä käytetään alan kirjallisuudesta tuttuja menetelmiä kuten esimerkiksi kiertomallia (Kellner et al, 1996) tai IDEAL-mallia (Gremba & Myers, 1997).

Ohjelmiston elinkaari (Software life-cycle) käsite on myös oleellista ymmärtää. Tämä tarkoittaa ohjelmiston elinkaarta ohjelmiston asiakasvaatimusten alkukohdasta ohjelmistotuotteen ylläpidon lopettamiseen (ISO/IEC 12207:1995, subclause 3.11).

4.2 Historia

Valoitetaan seuraavaksi hiukan ohjelmistoprosessien taustalla olevaa historiaa. Kuvassa 9 on aikajana, johon on merkitty joitakin ohjelmistoprosessien historian merkkipaaluja.

[1] http://www.sei.cmu.edu/sei-home.html
[2] http://www.sqi.gu.edu.au/spice/



Kuva 9 Ohjelmistoprosessit aikajanassa. Mukailtu: (Abrahamsson, 2002; Rautiainen, 2002)

Ohjelmistoprosessien tarpeen on katsottu alkavan noin 1960-luvun alussa. Sitä ennen ei ollut olemassa kovinkaan hyviä ohjelmiston laadun varmistamisen menetelmiä. Ensimmäiset prosessit ja ohjelmiston suunnittelumenetelmät olivat erittäin byrokraattisia (Highsmith, 1999). Niistä ehkä tunnetuin ajatusmalli jota käytetään edelleenkin paljon on vaiheittainen suunnittelu eli vesiputousmalli (Boehm, 1988).

Ohjelmistokokojen kasvaessa rakenteellisen ohjelmiston hallittavuusongelmia yritettiin ratkaista moduulaarisimmilla menetelmillä kuten olio-ohjelmoinnilla (Liberty, 1996). Olio-ohjelmointimetodi tarjosi paremmat edellytykset ohjelmistojen testaukselle ja helpotti esimerkiksi koko ohjelmiston hahmottamista. Koska olio-ohjelmointi, kuten C++ eroaa rakenteellisesta, kuten esimerkiksi C-ohjelmoinnista, on se vaikuttanut omalta osaltaan myös ohjelmistoprosesseihin. (Jaaksi et al, 1999; Abrahamsson et al, 2002).

1980-luvulla prosessien vaatima dokumenttimäärä oli jo valtava, koska pyrittiin määrittämään ja kuvaamaan kaikki mahdollinen. Näin uskottiin pystyttävän ratkaisemaan kaikki laatu- ja asiakasongelmat. Kehitettiin mm. Computer-Aided Software Engineering (CASE). Tästä on sitten myöhemmin 1990-luvulla saanut alkunsa myös Software Engineering Instituten (SEI)[1] kehittämä prosessinparannustekniikka (Software Process Improvement). (Highsmith, 1999.)

1990-luvun lopulla kehitettiin taas useita uusia radikaalistikin erilaisia metodeja tehdä laadukasta ja asiakkaiden vaatimuksia vastaavia ohjelmistoja. Uusissa virtauksissa pyrittiin vähentämään turhaa byrokratiaa ja samalla pyrittiin painottamaan ohjelmointityöhön liittyvän kommunikoinnin ja yhteistyön taitoja. Tämä voisi sanoa huipentuneen vuonna 2001 Agile Software Development -manifestiin[2]. Tässä vuonna 2001 järjestetyssä kokouksessa sen hetkiset "agile-ohjelmistoprosessigurut", kuten mm. Beck, Martin, Highsmith ja Cockburn, allekirjoittivat manifestin, jossa he lupasivat yhdessä kehittää uusia ja parempia tapoja tehdä toimivaa, laadukasta, muutettavaa ja asiakkaiden haluamaa laatua vastaavaa ohjelmistoa. (Cockburn, 2001.)

[1] http://www.sei.cmu.edu/sei-home.html
[2] http://www.agilemanifesto.org/

Prosessisimaiseen ajattelun nimeen vannoutuneiden lisäksi on aina ollut olemassa suunnittelijoita, jotka eivät usko prosesseihin, vaan pitävät niitä hidastavina tekijöinä. Tällaista tyyliä on kutsuttu myös nimellä "satunnainen ohjelmointi" (accidental programming). (Highsmith, 1999.) Kuten nimikin sanoo, tässä tyylissä ei mitään turhaa dokumentoida vaan annetaan ajatusten virran juosta ja eteen tulevien muutoksien tapahtua. Jos vesiputousmalli on selkeä top-down mallinen lähestymistapa, niin tämä satunnainen tapa on ilmiselvä down-top.

Tällä hetkellä eletään tilanteessa, jossa yritykset tekevät valintoja suunnitelmaohjautuvien sekä kevyiden agile-tyyppisten mallien keskellä. Organisaatiot näkevät ohjelmistokehitysmaailman polarisoituneena (Highsmith, 1999). Yritysten on nyt päätettävä haluavatko ne mennä "hallitun kaaoksen" suuntaan vai pysyä tutussa ja turvallisessa. Tämän paradoksin ratkaiseminen on myös yksi tämän työn haasteista.

4.3 Olio-ohjelmointi ja kehitysprosessit

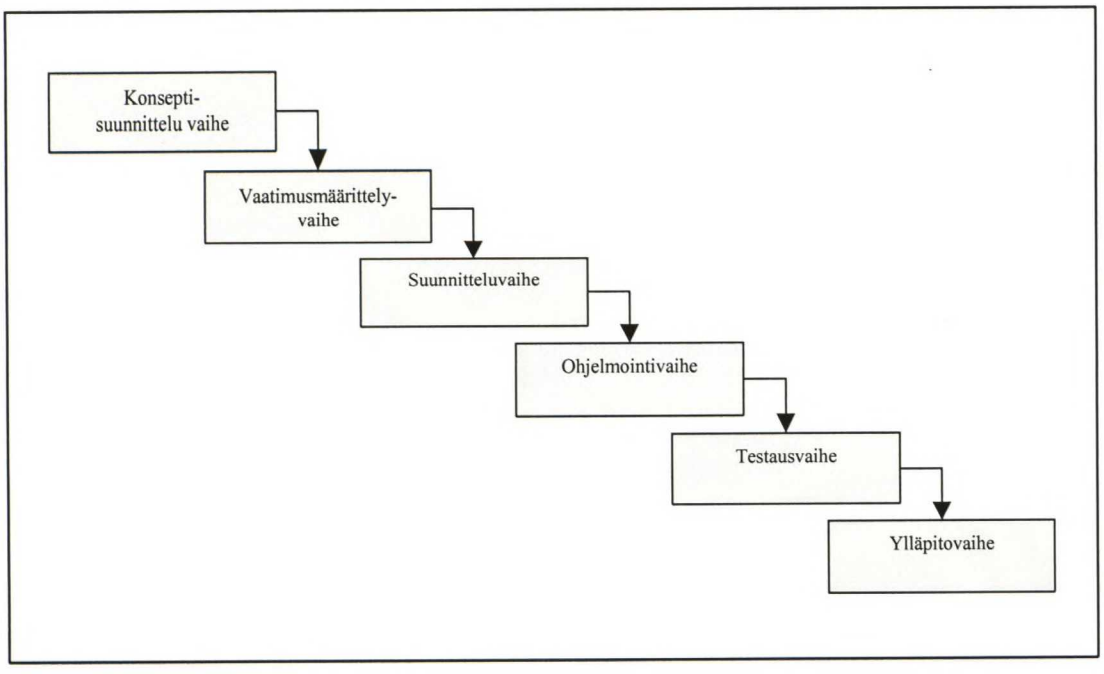
Olio-ohjelmointi ei sinänsä ole yksinään mikään ratkaisu ohjelmistojen suunnitteluun. Olio-ohjelmointia voidaan pitää lähinnä työkaluna, jolla ohjelmiston suunnittelua voidaan paremmin hallita ja hahmottaa. Olio-ohjelmointi on kuitenkin muuttanut ohjelmoinnin luonnetta ja on sitä kautta vaikuttanut ohjelmistoprosessien ja metodien luonteisiin. Esimerkiksi Nokialla on käytetyt Object Modeling Technique (OMT ja OMT++) -ohjelmiston suunnitteluprosesseja, jotka perustuvat nimenomaan olio-ohjelmointiin (Jaaksi et al, 1999). Toinen esimerkki on The Rational Unified Process (RUP), jonka suunnittelumetodi on varta vasten suunniteltu olio-ohjelmoinnille (Abrahamsson et al, 2002).

4.4 Suunnitelmaohjautuvat lähestymistavat

Suunnitelmaohjautuvilla (plan-driven) prosesseilla tarkoitetaan sellaisia prosesseja, jossa prosessiin on dokumentoitu ja etukäteen *suunniteltu* ohjelmistonkehityksen aikana pidettävät katselmukset, vaatimukset ja suunnittelu- sekä arkkitehtuurisuunnitelmat. Suunnitelmaohjautuvat metodit toimivat parhaiten silloin, kun vaatimukset ovat tiedossa etukäteen ja ne ovat suhteellisen stabiilit ohjelmiston elinkaaren ajan. Boehm myös toteaa, että suunnitelmaohjautuvalähtöinen kehittäminen on elintärkeää turvallisuutta vaativissa sulautetuissa ohjelmistoissa. (Boehm, 2002.)

4.4.1 Vesiputousmalli

Kuten edelläkin todettiin, on vesiputousmalli erittäin käytetty ja vanha suunnittelutapa. Sulautetun ohjelmiston suunnittelussa käytettävä vesiputousmalli voisi olla esimerkiksi kuvan 10 kaltainen.



Kuva 10 Vesiputousmalli. (Laplante, 1997)

Sana vesiputous tulee siitä, että suunnittelu on jaettu useaan eri osaan (Kuva 10) ja jokainen vaihe tulee suorittaa loppuun ennen kuin voi siirtyä seuraavaan vaiheeseen. Vesi ei siis virtaa "koskaan" ylöspäin. Tästä aiheutuvatkin vesiputousmallin edut ja heikkoudet. (Highsmith, 1999; Jaaksi et al, 1999; Laplante, 1997.)

Vesiputousmallin mukaan tehtävä ohjelmiston suunnittelu aloitetaan tuotteen täydellisellä määrittämisellä. Määrittämisvaiheen jälkeen suoritetaan ohjelmiston arkkitehtuurisuunnittelu ja tämän jälkeen se ohjelmoidaan. Kun ohjelma on valmis, toteutetaan testausvaihe sekä tehdään tarvittavat korjaukset. Lopulta testivaiheen päättymisen jälkeen ohjelmisto päätyy asiakkaalle ja siitä eteenpäin alkaa ylläpitovaihe tuotteen elinkaaren loppuun saakka. (Highsmith, 1999; Jaaksi et al, 1999; Laplante, 1997.)

Vesiputousmalli toimii hyvin vain sellaisessa tilanteessa, jossa on tiedossa mitä lopputuotteelta halutaan ja mihin sitä aiotaan käyttää. Todellisuudessahan asiakas- ja tekniset vaatimukset aina muuttuvat, joten vesiputousmalli ei ole ideaalisena kovinkaan käytännöllinen. Boehm toisaalta artikkelissaan toteaa, että pienillä muutoksilla kuten takaisinkytkennällä perättäisten vaiheiden välillä saadaan vesiputousmallia parannettua ja joitakin sen kankeudesta aiheutuvia ongelmia korjattua. Lisäksi ohjelmisto- ja laatukirjallisuudessa puhutaan myös ns. V-mallista, jossa pyritään käyttämään niin ikään takaisinkytkentöjä vesiputousmallin heikkouksien ja laadun parantamiseksi. (Boehm, 1988, 2002; Highsmith, 1999; Jaaksi et al, 1999; Laplante, 1997, Yli-Olli & Hokkanen, 1991.)

Soveltuvuus

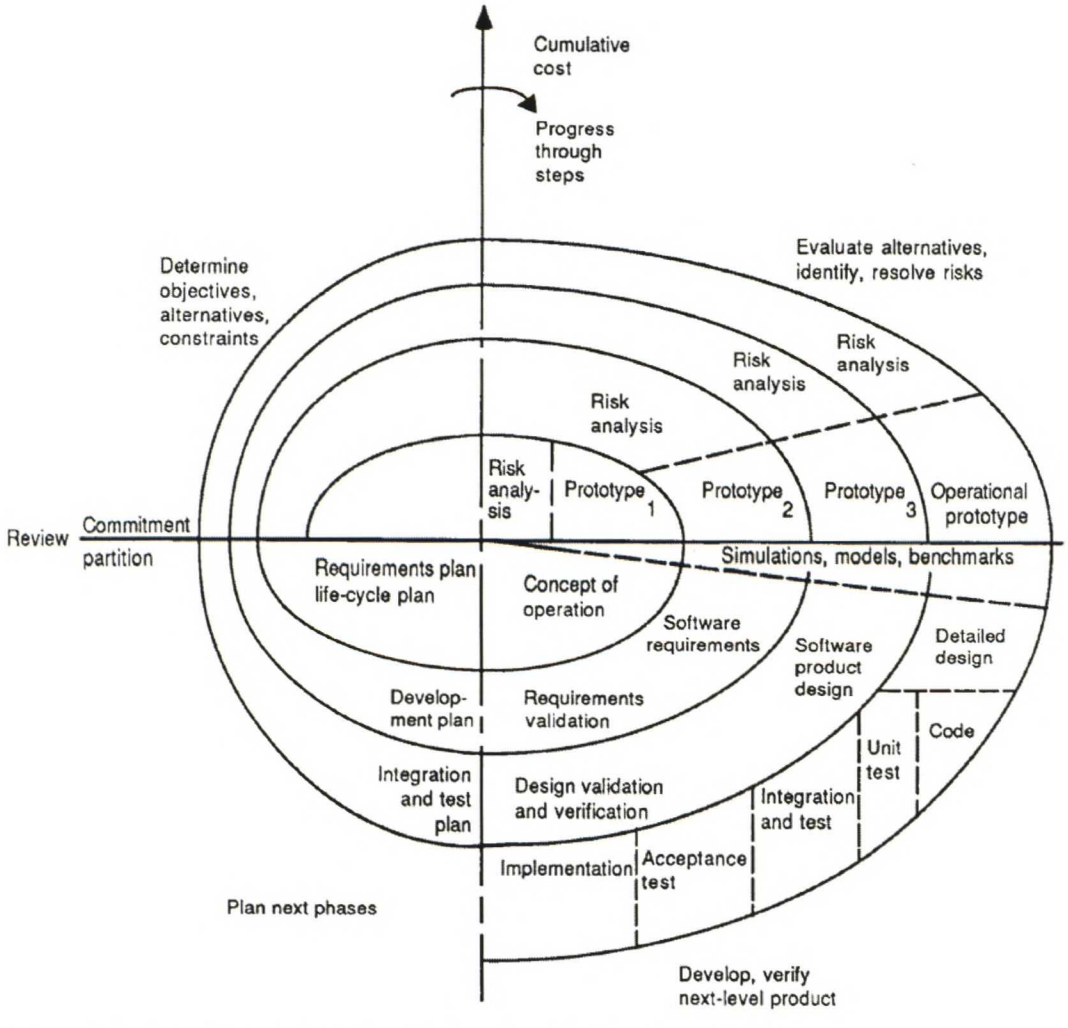
Sulautetun ohjelmiston suunnittelun kannalta vesiputousmallissa on kuitenkin tiettyjä etuja. Ideaalinen vesiputousmalli pakottaa esimerkiksi elektroniikkasuunnittelijat määrittelemään laitteiston ratkaisut etukäteen, ja koska laitteet testataan loppuvaiheessa, käytössä on yleensä aina uusin tuotantovalmis elektroniikka. Lisäksi esimer-

kiksi Vaisalan tapauksessa jotkut sulautetut ohjelmistot ovat suhteellisen pienikokoisia, joten niiden määrittäminen tarkasti etukäteen on mahdollista.

4.4.2 Spiraalimalli

Suunnitelmaohjautuvaa ohjelmistotuotantoa voidaan lähestyä myös toisella tavalla. Boehm esitti vuonna 1988 spiraalimallin, joka perustui voimakkaasti vesiputousmalliin. Spiraalimalli pyrkii kuitenkin myös yhdistämään muita ohjelmiston suunnittelukäytäntöjä kuten evoluutio- (evolutionary development) ja ohjelmoi-korjaa (code-fix) menetelmiä. (Boehm, 1988.)

Spiraalimallia kutsutaan myös riskiohjatuksi (risk-driven) prosessiksi (Pulli et al, 1994). Kuvassa 11 on esitetty Boehmin oma näkemys spiraalimallisesta prosessista. Malli on erittäin tunnettu alan kirjallisuudessa, joten siitä löytyy myös useita muunnoksia ja yksinkertaistuksia. (Boehm, 1988; Laplante, 1997.)



Kuva 11 Ohjelmiston kehityksen spiraalimalli. (Boehm, 1988)

Spiraalimallin perusajatus on se, että jokainen vaihe alkaa riskianalyysilla. Analyysin tuloksen perusteella kehittäminen keskeytetään tai sen annetaan jatkua. Spiraalimallissa painotetaan lisäksi prototyypitystä suunnittelun alkuvaiheessa. Prototyypitysten

ja riskianalyysien avulla pyritään selvittämään ja saamaan uutta tietoa asiakasvaatimuksista, tarvittavista ominaisuuksista sekä muista testien esiin tuomista ongelmista. Kun nämä ensimmäiset prototyypitys- ja riskianalyysivaiheet ovat ohi, käyttäytyy spiraalimalli kehityksen loppuvaiheessa vesiputousmallin tapaan. (Laplante, 1997; Boehm, 1988.)

Spiraalimalli on geneerinen malli, joka ei määrittele katselmuksien ajankohtaa kierroksien väleillä. Tämän takia yrityksen pitää itse muokata malli sopivaksi omaan katselmointikäytäntöönsä. (Pulli et al, 1994.)

Soveltuvuus

Malli sopii käytettäväksi sulautetun ohjelmiston suunnitteluun. Se tarjoaa prototyypitystyövaiheiden avulla mahdollisuuden kehittää sulautettua ohjelmistoa asteittain. (Pulli et al, 1994.) Sen avulla pystytään myös hyvin vastaamaan ohjelmiston vaatimusten muutoksiin suunnittelun alkuvaiheessa. Spiraalimallin heikkoutena voidaan kuitenkin pitää sen kykenemättömyyttä nopeisiin sekä suunnittelijatason päätöksiin, mutta toisaalta mallin vaatimat byrokraattisemmat ja ylemmän tason katselmukset tuovat prosessiin ulkopuolista asiantuntijuutta (Boehm, 2002).

4.5 Agile-tyyppinen lähestymistapa

Sana agile tarkoittaa sanakirjasta katsottuna ketterää, notkealiikkeistä, ripeäliikkeistä tai terävää [1]. Mutta mikä tekee ohjelmistoprosessista agilen? Abrahamsson et al. määrittävät, että ohjelmistoprosessi on agile, jos se on pienissä sekä nopeissa vaiheissa toteutettava (incremental), tuotteen asiakkaan kanssa tiiviissä yhteistyössä tehty (cooperative), yksinkertainen ja helposti omaksuttavissa (straightforward) ja hyvin muutokset huomioon ottava (adaptive). (Abrahamsson et al, 2002.)

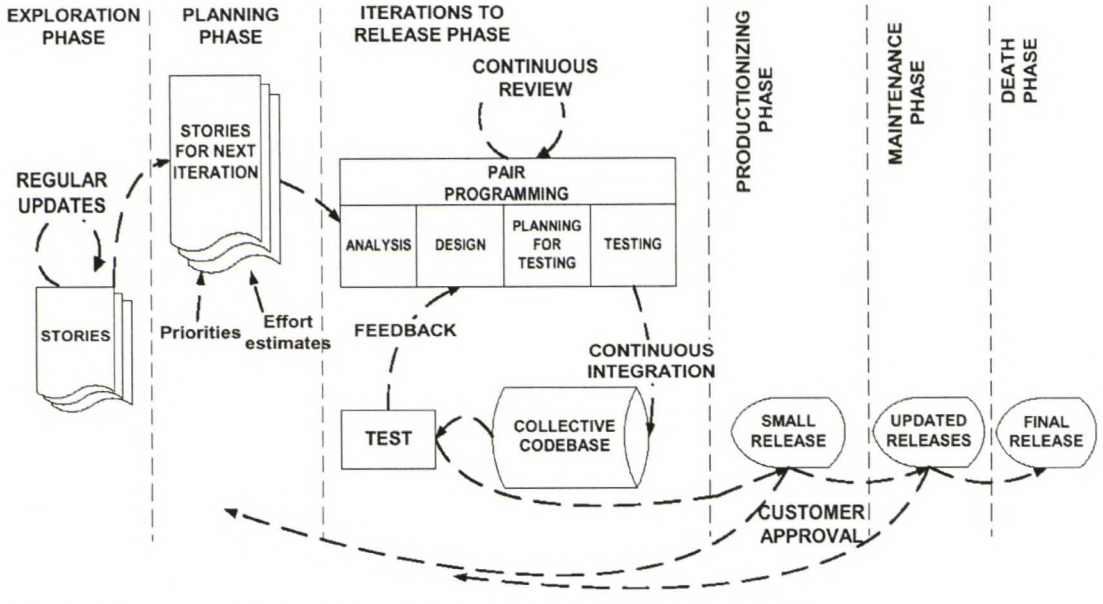
Ehkä tunnetuin agile-ohjelmointitapa on extreme programming (XP). Tämän kehitti vuonna 1999 Beck Kent (Beck, 2000). Tätä on pidetty myös agile-ohjelmointitapojen kehityksen alkuna. Lisäksi jälkikäteen on uudelleen löydetty tai kehitetty joitakin agile-tyyppisiä lähestymistapoja ohjelmistokehitykseen, kuten Crystal Methods, Scrum, Feature Driven Development, Lean Development tai Adaptive Software development. (Highsmith & Cockburn, 2001; Abrahamsson et al, 2002.)

4.5.1 XP

Extreme programmingin (XP) tarkoitus on lyhentää ohjelmiston suunnitteluaikaa paremmalla sekä "kevyemmällä" suunnitteluprosessilla. Ideana on, että tärkeintä on saada ohjelmisto ajallaan valmiiksi sekä suunnitella se parhaimpiin tunnettuihin käytännön kokemuksiin perustuen. (Abrahamsson et al, 2002; Beck, 2000.)

XP:n ohjelmiston elinkaari koostuu viidestä eri vaiheesta: exploration, planning, iterations to first release, productionizing, maintenance ja death (Beck, 2000). Vaiheet on esitetty kuvassa 12.

[1] MOT Electronic Library of Dictionaries. Kielikone Ltd. 1997



Kuva 12 XP prosessin elämänkaari. (Abrahamsson et al, 2002)

XP-prosessi alkaa tutkiskeluvaiheella (exploration phase). Tässä vaiheessa tutkitaan teknisiä mahdollisuuksia sekä kirjataan asiakkaan kertomuksia (stories) ylös niin paljon, että ollaan valmiit tekemään ainakin iteraatiovaiheen (iteration phase) ensimmäinen kierros. Suunnitteluvaiheessa (planning phase) päätetään asiakkaan kanssa mitkä kertomukset seuraavan iteraatiovaiheen aikana toteutetaan. Iteraatiovaiheet kestävät yhdestä neljään viikkoa. Tärkeää on huomioida, että ensimmäiseen iteraatiovaiheeseen pyritään tekemään/valitsemaan sellaiset kertomukset, jotka pakottavat suunnittelijat tekemään ohjelmiston arkkitehtuurin lähes kokonaan valmiiksi. Lopulta tuotantovaiheessa (productionizing phase) syklien pituutta lyhennetään ja samalla aletaan jäädyttelemään ohjelmistoa, jotta riskit loppua kohden pienenisivät. Tuotantovaiheen jälkeen tuote luovutetaan asiakkaalle ja samalla alkaa ylläpitovaihe (maintenance phase). Ylläpitovaiheessa korjataan virheitä sekä tehdään tarvittaessa uusia ominaisuuksia ohjelmistoon. Ohjelmiston elämänkaari päättyy tuotteen lopettamiseen. (Beck, 2000; Jeffries et al, 2001.)

Hyvin toteutettuna XP pystyy ottamaan huomioon alati muuttuvat asiakasvaatimukset. Tämän mahdollistaa lyhyet iteraatiovaiheet, prosessin takaisinkytkennät, asiakkaan aktiivinen osallistuminen sekä jatkuva ohjelmiston testaus ja integrointi. (Abrahamsson et al, 2002.)

XP pyrkii minimoimaan dokumentoinnin pariohjelmoinnilla. Periaatteena on, että jos kaksi ihmistä pystyy lukemaan koodia, niin silloin pystyy kolmaskin (Grenning, 2001). XP:n dokumentointi perustuukin suurimmalta osin juuri tähän hyvään lähdekoodin dokumentointiin. Tämä ei näin ollen välttämättä sovi muuhun tuotekehityskulttuuriin, kuten esimerkiksi Beck itsekin toteaa. (Beck, 2000; Grenning, 2001; Jeffries et al, 2001.)

Soveltuvuus

XP:n käytöstä sulautettujen ohjelmistojen suunnittelussa löytyy joitakin olio-ohjelmointiin perustuvia käytännön esimerkkejä kuten Grenningin artikkeli (Gren-

ning, 2002). XP heikkoudeksi voisi lukea Vaisala Instrumentsin tapauksen kannalta olio-ohjelmointiriippuvuuden. Tämä siksi, että ohjelmistolle pitäisi XP:n mukaan tehdä koko ajan automaattitestauksia, jotka ovat erittäin vaikeasti toteutettavissa C- tai assembler-ohjelmoinnin avulla.

4.5.2 Evo

Evo-metodi (Evolutionary Project Management methods) ei puhdasoppisesti ole agile, mutta se sisältää paljon agile-tyyppisille metodeille tuttuja ominaispiirteitä ja siksi olen sen tähän lukuun sijoittanut. Evo-metodi halutaan erottaa inkrementaalisesta tai perinteisestä evoluutiomaisesta ohjelmiston suunnittelusta. Ne eroavat toisistaan esimerkiksi siten, että Evo-metodi sisältää ajattelutakaisinkytkentöjä, nopeamman suunnittelun aloituksen ja strategisen suunnittelun, kun taas tavallinen inkrementaalinen tai evoluutiomainen suunnittelu on vain lyhyiden vesiputousmallien peräkkäistä toistoa. Kuvan 13 prosessi kuvaa tavallista evoluutiomaista kehitystä, kun taas kuvan 14 prosessi Evo-tyyppistä. Kuvat pyrkivät selventämään prosessien eroja. (Malatouxin, 2003; Gilb, 1997.) Lisäksi voisi todeta, että Abrahammson et al. ovat kirjassaan maininneet Evon yhdeksi XP:n juurista (Abrahamsson et al, 2002).

Incremental Development Model

Complete Detailed Frozen	Complete Detailed Frozen	Build/test	Build/test	Build/test	Build/test	Build/test	
Requirements Analysis & specification	**Design Specification**	**Step 1** →	**Step 2** →	**Step 3** →	**Step 4** →	**Step n** →	**Acceptance Test**

Kuva 13 Evoluutiomainen ohjelmiston kehitysmalli. (Gilb, 1997)

Head-and-Body Evo Model

Head	Head	Body	Body	Body	Body	Body	Head
		←experience	←	←	←	←	
Requirements Analysis & specification	**Design specs**	**Step 1** →	**Step 2** →	**Step 3** →	**Step 4** →	**Step n** →	**Acceptance Test**
Needs →	**Ideas** →	**Micro-project**	**Micro-project**	**Micro-project**	**Micro-project**	**Micro-project**	**Product ship**

Kuva 14 Evo-tyyppinen ohjelmiston kehitysmalli. (Gilb, 1997)

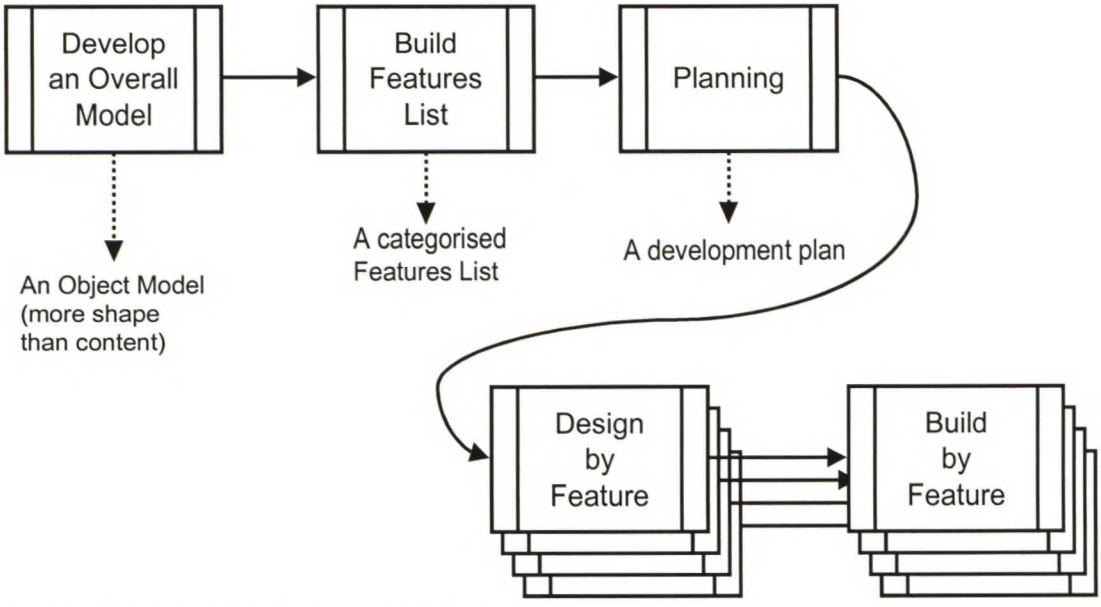
Evo-suunnitteluprosessissa on kaksi eri osaa, eli "pää-" ja "lihasosa". Päässä tapahtuu koko prosessin ajattelu ja muutosten toteutus. Lihasosassa puolestaan suoritetaan valmiiksi ajatellut osaset mikroprojekteina. Perusajatus projektin läpiviemisessä on se, että alussa ei käytetä turhaa energiaa liian raskaisiin ja yksityiskohtaisiin vaatimusten määrittelyihin. Vaatimuksia tarkennetaan ja muutetaan joustavasti projektin eri vaiheiden välillä, kun todellista tietoa vaatimusten pohjalle on saatavissa. (Gilb, 1997.)

Soveltuvuus

Evo-metodi on hyvin samantyylinen kuin XP, mutta koska Evon tekijä on eri henkilö, löytyy prosessin dokumentoinnista ja jaksottamistavoista eroja. Näitä ei ole syytä lähteä tässä työssä erittelemään. (Malatouxin, 2003; Gilb, 1997; Beck, 2000.) Evon käytöstä ja sen hyödyistä sulautetun ohjelmiston suunnittelussa on vaikeaa löytää puolueetonta tietoa, koska tarjolla oleva kirjallisuus on erittäin konsulttipainotteista. Evo-metodin käytöstä sulautetun ohjelmiston suunnittelussa saatiin kuitenkin tietoa benchmarkingin yhteydessä, koska toinen benchmarking-kumppaneista ilmoitti käyttävänsä Evo-metodeja oman sulautetun ohjelmiston suunnittelunsa taustalla.

4.5.3 Feature Driven Development

Feature Driven Development (FDD)-tyyppisen lähestymistavan ei ole tarkoitus kattaa koko ohjelmiston elinkaarta, vaan se keskittyy elinkaaren suunnittelu- sekä toteutusvaiheeseen. FDD onkin suunniteltu muuta ohjelmiston elinkaarta tukevaksi metodiikaksi. FDD:n suunnittelu- ja toteutusvaihe koostuu viidestä eri vaiheesta, jotka on esitetty kuvassa 15.



Kuva 15 FDD prosessi. (De Luca, 2003)

FDD:n mukaisen ohjelmiston suunnittelun ensimmäinen vaihe (Develop an Overall Model) alkaa kertomusten (use cases) ja teknisten vaatimusten keräämisellä. Kun niitä on riittävästi kassassa siirrytään ominaisuuslistan (feature list) tekoon. Tässä vaiheessa vaatimukset puretaan pienempiin osiin eli ohjelmiston ominaisuuksiksi (features). Suunnitteluvaiheessa (planning) pääsuunnittelijat kokoavat ominaisuuksista sopivia ominaisuuspaketteja (set of features), jotka määrittävät iteraatiovaiheiden työt. Suunnittelun jälkeen siirrytään ominaisuussuunnittelu- ja ohjelmointivaiheisiin. Nämä vaiheet kestävät muutamasta päivästä pariin viikkoon. Huomioitavaa on myös, että eri ominaisuuspaketteja (set of features) voidaan suunnitella yhtäaikaisesti. (Abrahamsson, 2002; De Luca, 2003.)

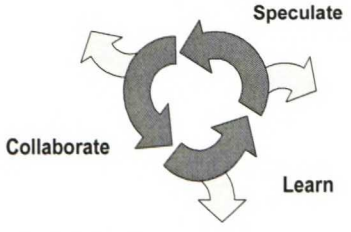
Soveltuvuus

FDD:stä ei ole tarjolla tällä hetkellä muuta kuin konsulttikirjallisuuden tuotoksia, joten sen käytöstä sulautetun ohjelmiston tuotannosta tai edes sovellusohjelmistotuotannossa ei ole totuuspohjaista näyttöä. (Abrahamsson et al, 2002.) FDD:ssä voisi kuitenkin olla ainesta tuoteprosessin aliprosessiksi, koska se on jo alunperinkin suunniteltu tukiprosessiksi. Lisäksi FDD:n lähestymistavan etu on se, että sen on väitetty soveltuvan turvallisuutta vaativien ohjelmien suunnitteluprosessiksi (Abrahamsson, 2002).

4.5.4 Adaptive Software Development

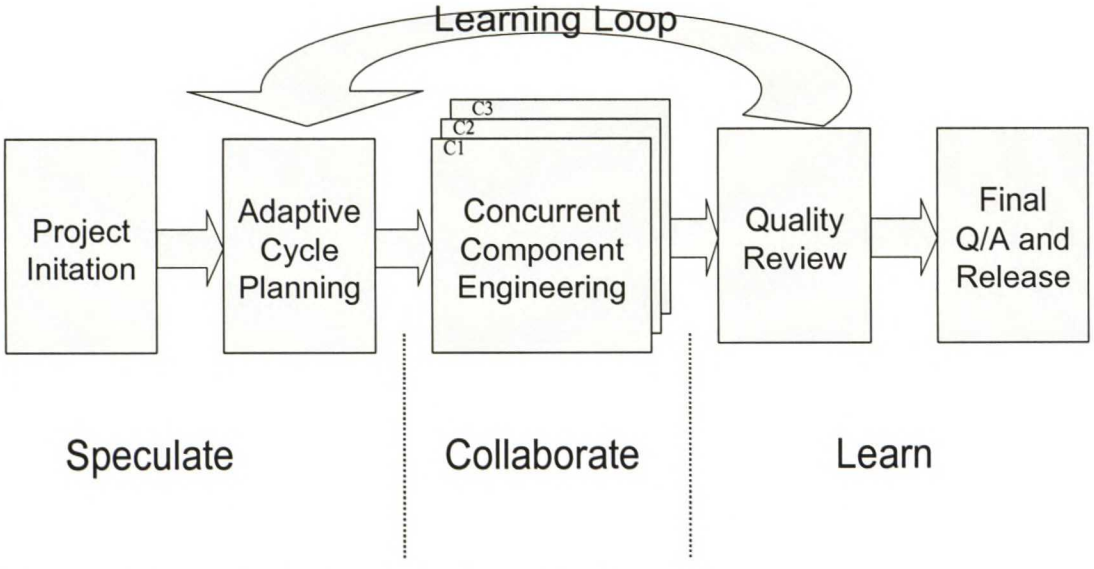
Adaptive Software Development (ASD) on vuonna 2000 Highsmithin julkaisema metodi, joka polveutuu hänen aikaisemmin kehittämistään iteratiivisista suunnittelumenetelmistä. ASD on pääasiassa tarkoitettu suurien ja kompleksisten systeemien suunnitelumetodiksi. ASD rohkaisee inkrementaaliseen ja iteratiiviseen suunnitteluun sekä samanaikaiseen prototyypitykseen. (Highsmith, 1999; Abrahamsson et al, 2002.)

ASD prosessi koostuu sykleistä, jotka ovat jakautuneet kolmeen eri vaiheeseen. Syklin eri vaiheet ovat graafisesti havainnollistettu kuvassa 16.



Kuva 16 ASD-sykli. (Highsmith, 1999)

Kuva 17 puolestaan selventää, kuinka syklit toimivat projektin aikana. Projekti lähtee liikkeelle alustuksella (project initation). Tässä vaiheessa projektille määritellään missio, jonka sisältämä tärkein sanoma on dokumentoitu projektivisioon (project vision charter), projektin datalehdeen (project data sheet) sekä tuotespesifikaatiohahmotelmaan (product specification outline). Alustusvaiheessa määritellään myös projektin aikataulu sekä sykleissä toteutettavat asiat. Highsmith myös erityisesti painottaa, että ASD on komponenttiorientoitunut. Tällä tarkoitetaan sitä, että on tärkeämpää saada syklin aikana tulosta aikaan, kuin vain suorittaa sille määritetyt tehtävät. Jokainen suunnittelusykli päättyy oppimis-takaisinkytkentään (Learning loop). Tässä vaiheessa pidetään laatukatselmus, jossa oleellisena tekijänä on aina mukana asiakkaan edustus. Viimeisenä vaiheena on laatutarkastus (Final Q/A and Release). Jos se läpäistään, voidaan tuote luovuttaa asiakkaalle. (Highsmith, 1999; Abrahammson et al, 2002.)



Kuva 17 ASD-metodin elämänkaari. (Highsmith, 1999)

Soveltuvuus

ASD ehdottaa erittäin vähän suoraan käytäntöön toteutettavia asioita päivittäiseen ohjelmistotuotantoon. Tärkeimmät asiat, jotka ASD-tyylisen kehityksen aikana tulisi toteuttaa, ovat iteratiivinen kehitys, komponenttipohjainen suunnittelu sekä asiakaskatselmuksien käyttö. ASD ei ehdotakaan, mitä pitäisi tehdä, vaan antaa lähinnä ohjeita miten asiat voisi tehdä. Tietoa ASD:n todellisesta hyödystä ohjelmistotuotannossa ei myöskään ole paljoakaan tarjolla. ASD:n suurin anti sulautetun ohjelmiston suunnittelulle onkin ehkä taustalla oleva ajatusmaailma sekä näkemys, kuinka voidaan hallita kompleksisia systeemejä. (Abrahamsson et al, 2002.)

4.6 Muut ohjelmiston kehitysprosessit

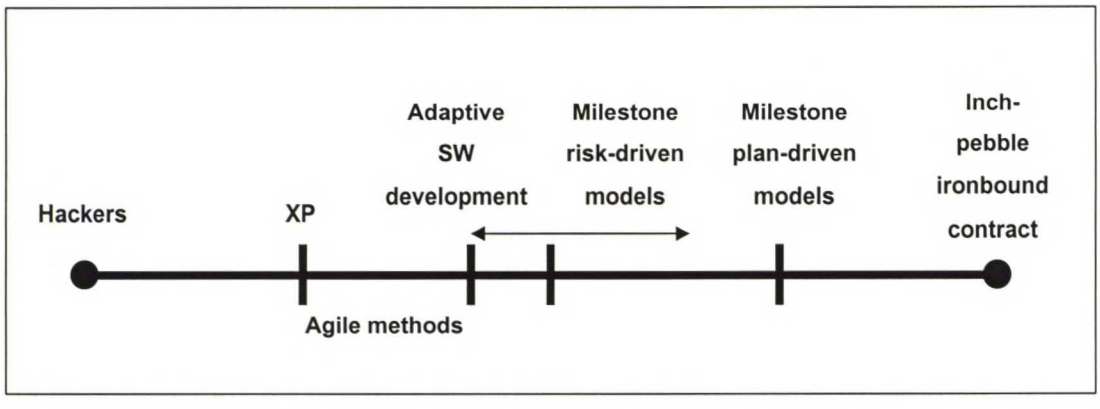
Tässä luvussa on nyt esitelty kahden tyyppistä lähestymistapaa ohjelmistojen kehitysprosesseihin, suunnitelmaohjautuvia sekä agile-tyyppisiä. Tähän lukuun valitsin esiteltäväksi vain sellaisia metodeja, joita pidän merkityksellisinä tämän työn kannalta. Kaikkien näiden tapojen lisäksi on olemassa lähes rajaton joukko muita ohjelmiston kehitysprosesseja. On olemassa esimerkiksi ns. sattumaan perustuvia tai hakkerimenetelmiä. Nämä tavat tehdä ohjelmistoa ovat erittäin vapaita ja niissä ei käytetä hyväksi mitään kuvattua tai kiinteätä prosessia, joten ne soveltuvat huonosti laatustandardeihin ja siten teollisuuden käyttöön. Kirjallisuudesta löytyy myös tietyn tyyppisiin esimerkiksi turvallisuutta vaativien sulautetun ohjelmiston (Korhonen & Kalaoja, 1992) tai muutoksen hallintaan (Mäkäräinen, 2000) suunnattuja prosessien ja toimintaohjeiden parannusehdotuksia. Nämä eivät kuitenkaan tarjoa kattavaa mallia koko ohjelmiston elinkaaren pituiselle prosessille.

Agile-tyyppinen ohjelmiston kehitys on viime vuosien uusin suuntaus. Tässä työssä esiteltiin neljä erilaista agile-metodia, mutta niitä on tunnustetusti olemassa paljon enemmän. Esimerkiksi Crystal methods, Lean Development, Open Source Development, Scrub, The Rational Unified Process ja Dynamic Systems Development Met-

hodology ovat maailmalla tunnettuja ja tunnustettuja agile-metodeja. Yhteistä kaikille agile-metodeille on perusajatus saada koodi nopeasti tehtyä sekä samalla toimivaksi. Lisäksi ne painottavat asiakas- ja tiimityölähtöisyyttä. (Highsmith & Cockburn, 2001.)

4.7 Yhteenveto ohjelmistoprosesseista

Parhaiten koko ohjelmiston kehitysprosessien "toimintakentän" pystyy hahmottamaan Boehm piirtämällä suunnitelmallisuusspektrillä (Kuva 18) (Boehm, 2002). Kuvassa vasemmassa laidassa ovat erittäin kevyet hakkerityyppiset metodit, kun taas kuvan oikeassa laidalla ovat (Inchpebble ironbound contract) millintarkasti johdetut ja määritellyt metodit. Kuvasta nähdään, että XP sekä muut agile-metodit sijoittuvat luonteeltaan enemmän vasemmalle puolelle eli vapaamman suunnittelun suuntaan. Spiraalimainen riskiohjattu (risk-driven) suunnittelu sijoittuu puolestaan noin puoleen väliin spektriä. Suunnitelmaohjautuvat metodit taas ovat enemmän oikealla, riippuen niiden suunnittelun ohjauksen yms. tarkkuudesta.

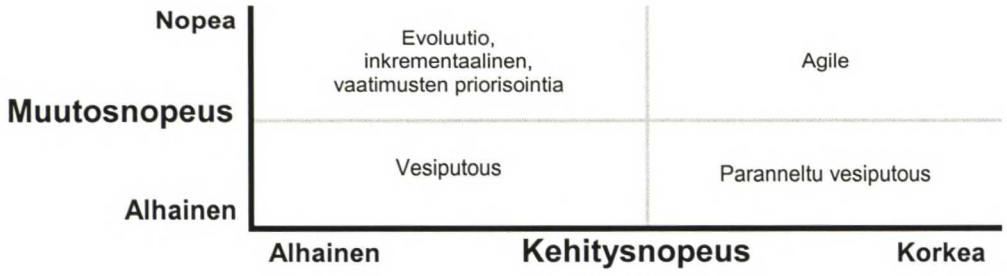


Kuva 18 Ohjelmiston suunnitelmallisuusspektri. (Boehm, 2002)

Kuva 18 antaa vaikutelman, että suunnitelmaohjautuvat- ja agile-metodit ovat aivan eri maata. Toisaalta tilanne ei ole näin mustavalkoinen. Kummankin mallin tulisi esimerkiksi perustua asiakaslähtöisyydelle. Lisäksi kumpikin metodi suorittaa alussa etukäteissuunnittelua, mutta tässä agile-metodit painottavat enemmän itse suunnitteluprosessia kuin siitä syntyvää dokumentaatiota (Boehm, 2002). Näkyvin ero näiden metodien välillä on suunnittelu ja toteutusvaiheessa. Agile-metodit painottavat enemmän toimivaan loppuratkaisuun pääsemistä iteratiivisesti sekä joustavilla keinoilla ja lisäksi asiakkaan sekä yrityksen sisäistä kommunikointia. Kuten jo useaan otteeseen mainittu, tällä toiminnalla agile-metodit pystyvät vastaamaan kompleksisemman organisaation haasteisiin, kuten jatkuviin vaatimusten muutoksiin. (Highsmith & Cockburn, 2001.)

Mikä olisi sitten paras metodi mihinkin tilanteeseen? Tähän ei ole yhtä oikeaa vastausta, vaan kaikki riippuu tilanteesta ja siitä mihin metodia tarvitaan (Boehm, 2002). Nopeaa kehitystä sekä muutosta vaativiin projekteihin voitaisiin sanoa tarvittavan agile-metodeja, kun taas suunnitteluihin, joissa asiakasvaatimukset ja lopputuotteen ominaisuudet ovat etukäteen suhteellisen hyvin tunnettuja, sopisi yhtä hyvin vesipu-

tousmalli. Kuvan 19 nelikenttä pyrkii havainnollistaa mallien soveltuvuuksia eri tyyppisiin tilanteisiin (Highsmith, 1999; Rautiainen, 2002).



Kuva 19 Ohjelmiston kehitysmetodimatriisi. Mukailtu: (Highsmith, 1999; Rautiainen, 2002)

Valitsee yritys sitten minkä tahansa lähestymistavan, joutuu se kumminkin loppujen lopuksi sovittamaan valitsemansa metodin tai metodit yrityksen omaan kulttuuriin sekä muihin prosesseihin sopiviksi. Yksiselitteistä ratkaisua ei siis ole, joten sulautetun ohjelmiston suunnittelussa voidaan käyttää yhtä hyvin agile-metodeja tai tavallisempaa vesiputousmallia. (Grenning, 2002; Pulli et al, 1994; Laplante, 1997.) On syytä kuitenkin pitää mielessä, että sulautetun ohjelmiston suunnittelu vaatii enemmän ja erilaisia vaatimusmäärittelyjä, kuin mitä esimerkiksi Microsoft Windows-ympäristössä toimivat sovellusohjelmistot (Ronkainen et al, 2002). Lisäksi Vaisala Instrumentsin tapauksessa suunnitteluprosessilta vaaditaan turvallisuusstandardien mukaisten suunnitteluprosessien tukemista, jota kaikki agile-metodit eivät tunnetusti tee (Boehm, 2002).

Nykyisin on myös yleistä suunnitelmalähtöisen sekä agile-metodin yhdistäminen sopivilta osin esimerkiksi Grenning (2001) ja Fowler (2001). Toisin sanoen, yhdistelmämalleissa yritetään poimia parhaat palat kummastakin tyylistä. Tällaista lähestymisnäkökulmaa on Boehmin mukaan esimerkiksi Highsmith käyttänyt hyväksi edelläkin esitellyssä ASD-metodissa. (Highsmith, 2002; Boehm, 2002.)

Vaisala Instrumentsin sulautetun ohjelmistojen koodirivien määrä on kasvamaan päin. Jos historiasta on mitään opittavaa, niin ainakin se, että ohjelmistokoon kasvaessa on suunnittelun hallinta aina vain vaikeutumassa sekä riskien ja muutosten määrä lisääntymässä (Boehm, 1988, 2002). Tässä on siis todellinen haaste kehityshankkeellemme.

5 OHJELMISTOSTANDARDIT JA LAATU

Luvussa 4 on käyty läpi erilaisia ohjelmiston kehitysmetodeja. Laadun kannalta valittava metodi ja sen pohjalta tehty prosessikuvaus on yritykselle tärkeä (Lecklin, 2002). Uusimmatkaan metodit eivät ole kuitenkaan pystyneet puolueettomasti todistamaan niiden todellista taloudellista hyötyä tai ohjelmiston laadullisia parannuksia. Lukuarvoissa puhutaan keskimäärin ehkä noin 13% tuottavuuden lisäyksestä ja 75% laadun parantumisesta 12 vuoden kuluessa vuoden 1985 jälkeen. (Glass, 1999.) Toisaalta esimerkiksi Charles C. Mann artikkelissaan väittää, että ohjelmiston laatu on ennemminkin heikkenemään päin ja tämän syyksi hän nimeää esimerkiksi nykyiset uudet ohjelmointityylit sekä ohjelmistoyritysten laatukriteerien löysentymisen. Laatuasioissa on joustettu ja on päästetty tietoisesti asiakkaalle laitteita, joissa tiedetään olevan virheitä. Tästä tuorein esimerkki on Windows XP, joka teki tämänhetkisen korjauksen nopeusennätyksen julkaisemalla ohjelmiston julkaisupäivänä verkkosivuillaan 18 megatavun verran korjaustiedostoja. (Mann, 2002.)

Yritykset todistavat tuotteidensa laadun sekä luovat imagoaan yleensä johonkin laatustandardiin tukeutuen (Hannus, 1994). Vaisalalla on esimerkiksi käytössä ISO 9001 sekä ympäristöstandardi ISO 14001. ISO 9001 on ohjelmiston kannalta oleellisin, joten seuraavassa käsitellään tarkemmin sen ohjelmistolle määrittämiä ohjeistuksia. Lisäksi esitellään muita tämän työn kannalta tärkeitä ohjelmistotuotannon standardeja. Tässä osiossa ei tulla kuitenkaan esittelemään ohjelmistoprosessien arviointiin ja kehittämiseen suunnattuja standardeja kuten ISO/IEC 15504 (SPICE).

5.1 ISO 9001

ISO 9001 kuuluu ISO 9000-sarjaan. Se on tarkoitettu organisaatioille, joiden toimintaan sisältyy suunnittelua, tuotekehitystä, tuotantoa, asennusta ja toimituksen jälkeisiä palveluja. ISO 9001:n pohjalta on tehty ISO 9000-3, joka tulkitsee ISO 9001:a ohjelmistotuotannon kannalta, koska ISO 9001 ei anna kovinkaan konkreettisia tai ohjelmistojen tekemiseen liittyviä ohjeita. (Yli-Olli & Hokkanen, 1991; ISO 9001:2000.)

5.2 ISO 9000-3

ISO 9000-3 on siis tärkein standardi tämän työn kannalta, joten käydään sitä läpi hiukan tarkemmin. ISO 9000-3 antaa ohjeita ISO 9001:n noudattamiseksi, mutta se ei kuitenkaan määrittele yksityiskohtaisesti ohjelmistoa tekevän organisaation tuotekehitysprosessia ja lopputuotteelta vaadittavia ominaisuuksia eikä anna yksityiskohtaisia ohjeita laatujärjestelmälle. Toisin sanoen standardia käyttävän on itse tulkittava ja löydettävä sitä vastaava toimintatapansa. (ISO 9000-3:1997; Yli-Olli & Hokkanen, 1991.)

ISO 9000-3 jakautuu kolmeen pääosaan: puitteisiin, projektin elinkaareen ja tukitoimintoihin (ISO/IEC 12207:1995). Laatujärjestelmän puitteilla tarkoitetaan johdon vastuuta, mm. asiakkaan määrittämiä hyväksymiskriteerejä sekä toimintatapoja, dokumentoitua laatujärjestelmää, laatuauditointeja sekä korjaavien toimenpiteiden määrittämistä. ISO 9000-3 ei ota kantaa siihen mitä ohjelmiston elinkaarimallia käyte-

tään. Kuitenkin elinkaaren vaiheista kuten ostajan tarpeiden määrittelystä, kehitystyön suunnittelusta, testauksesta ja validoinnista sekä ylläpitovaiheesta, on standardissa annettu määräyksiä. Tukitoiminnoilla ISO 9000-3 tarkoittaa tehtäviä, joita ei voi kohdistaa tiettyyn toimituksen vaiheeseen. Tällaisia ovat mm. versionhallinta, dokumenttien valvonta, laatutiedostojen ylläpito, laadun mittaamisen työvälineet ja menetelmät sekä alihankinnat. Tukitoiminnoissa korostetaan erityisesti versionhallintaa. Versionhallinnan järjestelmän tulee mm. yksilöidä koko ohjelmisto ja sen kunkin osan virallinen versio. Muutospyynnöt on lisäksi voitava tunnistaa ja jäljittää ehdotuksesta käsittelyn loppuun saakka. (Yli-Olli & Hokkanen, 1991; ISO/IEC 12207:1995.) Toimittajan on myös tehtävä versionhallintasuunnitelma, joka sisältää toimintatapojen kuvaukset jolla hallintaa tehdään, työkalujen määrittämiset, aikatalutukset, vastuukysymykset sekä näiden liittyminen suunniteluun ja ylläpitovaiheeseen (ISO/IEC 12207:1995 subclause 6.2.1.1).

5.3 ISO/IEC 12207

ISO/IEC 12207 (uusin lisäys: ISO/IEC 12207:1995/Amd.1:2002) on ISO-standardi ohjelmistojen elämänkaari prosesseille. Se pyrkii antamaan kehykset ohjelmistojen elämänkaarille. Se ei ota kantaa käytettävään suunnitteluprosessiin, mutta se tarjoaa prosessin jolla ohjelmistoprosesseja voidaan määrittää, kontrolloida sekä parantaa. ISO/IEC 12207 on ISO 9000-3:a tarkempi kuvaus ohjelmistoprosessien elinkaaresta. Tämän takia ISO 9000-3 viittaa paljon ISO/IEC 12207 standardiin. Voisi sanoa, että ISO/IEC 12207 on tietyiltä osin ISO 9000-3:n lukuohje samoin kuin ISO 9000-3 on ISO 9001 standardille. (ISO/IEC 12207:1995; ISO 9000-3:1997.)

5.4 ISO/IEC TR 15271

ISO/IEC TR 15271 on itse asiassa tekninen raportti. Sen tarkoitus on tarjota ohjeita ISO/IEC 12207:n noudattamiseksi. Se pyrkii yksityiskohtaisemmin erottelemaan asioita joita pitää ottaa huomioon noudatettaessa ISO/IEC 12207 standardia. Lisäksi se tarjoaa vaihtoehtoja, kuinka ISO/IEC 12207 standardia voidaan soveltaa. ISO/IEC TR 15271 ei ole kuitenkaan tarkoitus antaa kaiken kattavia selityksiä ISO/IEC 12207:n vaatimuksista. (ISO/IEC TR 15271:1998.)

5.5 IEEE 829

Standardi IEEE 829 on ohjelmiston testauksen dokumentoinnin standardi. Se pyrkii kuvaamaan tavallisimmat ohjelmiston testausdokumentit. Lisäksi se määrittelee niiden muodon ja sisällön. Standardi ei kuitenkaan määrittele mitä testausdokumentteja ohjelmistolta vaaditaan. (IEEE Std 829-1998.)

5.6 Ohjelmistojen turvallisuusstandardit

Edellä esitellyt standardit liittyivät lähinnä yleiseen laatuun ja sen asettamiin vaatimuksiin. Näiden lisäksi on olemassa joukko turvallisuusstandardeja, jotka asettavat ohjelmistoille lisävaatimuksia. Nämä standardit voidaan jakaa niiden kattavuuden perusteella kahteen osaan: suppeampiin sekä kaiken kattaviin. Esimerkiksi IEC

61508 & BS EN 61508:2002 pyrkii kattamaan kaikki osa-alueet, kun taas esimerkiksi EN 50271:2001 kattaa vain hapen, tulenarkojen sekä myrkyllisten kaasujen ohjelmistojen turvallisuusvaatimukset. (McDermid & Pumfrey, 2001.)

Tämän työn kannalta merkittävin turvallisuusstandardi on EN 50271:2001. Tämä standardi vaatii ohjelmistolta mm. sen, että asennettu ohjelmistoversio on aina tunnistettavissa, käyttäjän ei ole mahdollista muuttaa ohjelmiston funktioita sekä ohjelmisto on tehtävä struktuuriseksi ja modulaariseksi testauksen ja ylläpidon helpottamiseksi. Työn alkuvaiheessa on tutustuttu myös IEC 61508 standardiin, mutta koska se pyrkii kattamaan kaiken mahdollisen, on sen koko valtaisa, joten sen tulkintaan on varattava enemmän aikaa. Tämän takia standardia IEC 61508 ei käsitellä tämän kehityshankkeen aikana.

5.7 Laatujohtaminen

Jotta edellä esitetyistä standardeista olisi jotain hyötyä, on niiden vaatimukset pystyttävä täyttämään sekä lyhyellä että pitkällä tähtäimellä. Tähän pääsemiseksi vaaditaan yritykseltä pitkäjänteistä laatuajattelua sekä laatujohtamista.

Johdon on pystyttävä viemään laatuajattelu yrityksen läpi perusarvoista lähtien. Itse laatujohtaminen on samantapaista toimintaa, kuin mikä tahansa muukin yrityksen johtaminen. Laatujohtaminen alkaa johdon vision luomisella. Vision pohjalta johto tekee yritykselle mission, jossa se määrittelee laadullisen toiminnan päämääränsä tai tarkoituksensa vision toteutuksesta. Mission toteutukseen laaditaan strategia. Strategia asettaa suuntaviivat ja kehykset, jotka ohjaavat operatiivista toimintaa ja esimerkiksi ohjelmistoprosesseja. Kaiken tämän lisäksi johdon pitää asettaa laatutavoitteita, esimerkiksi ohjelmiston luotettavuus, asiakastyytyväisyys tai toimitusvarmuus. Tavoitteet voivat olla kovia tai pehmeitä, mutta ne on pyrittävä valitsemaan siten, että niiden avulla voidaan ohjata sekä seurata yrityksen toimintaa. (Lecklin, 2002.) Laatu ja sen johtaminen on hyvä pitää mielessä ohjelmistoprosessia kehitettäessä.

6 TUOTEKEHITYSPROSESSIEN JOHTAMINEN

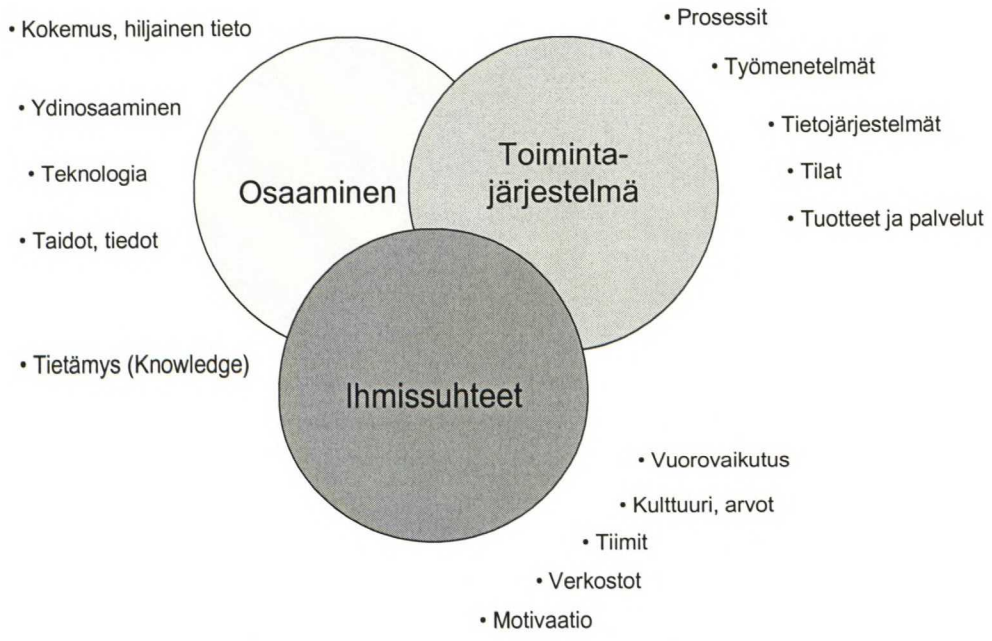
Tässä luvussa keskitytään prosessijohtamiseen sekä sen vaatimuksiin tuotekehitysjohtajan ja erityisesti ohjelmiston suunnittelusta vastaavan johtajan kannalta. Tämä on tärkeää prosessien kehittämisen kannalta, koska suunnitellut ja toteutetut prosessit pitää olla johdettavissa. Hyvin johdettavaa prosessia on myös vaikea suunnitella, jos ei tiedä johtamisen taustalla toimivaa ajatusmaailmaa (Laamanen, 2001).

Käsittely aloitetaan määrittelemällä prosessijohtamisen teoriaa yleisesti. Sen jälkeen analysoidaan ohjelmistosuunnittelumetodiikkojen korostamaa johtajan roolia prosessijohtamisajatteluun pohjautuen. Lisäksi kerrotaan lyhyesti prosessijohtamisen yhdestä tärkeimmästä työkalusta eli prosessimittarista. Lopuksi tutkitaan hiukan prosessimaailmaa vallan näkökulmasta, koska loppujen lopuksi prosessien perimmäinen tarkoitushan on toimia yrityksen johdon vallan välineenä ja kuvaajana.

6.1 Prosessijohtamisen määritelmä

Tuotekehitysprosessit kuvaavat tuotekehityksen toimintaa. Prosessiajattelun myötä on näkemys organisaatiosta kääntynyt funktionaalisen pystysuorasta vaakasuoraksi. Organisaatiot ovat muuttuneet toimintojen ja osaajien verkostoksi, jossa korostuvat sisäiset sekä ulkoiset asiakkaat. Tässä työssä kehitettävän ohjelmiston suunnitteluprosessin sisäisenä asiakkaana voidaan pitää esimerkiksi tuoteprosessia ja ulkoisena asiakkaana lopullisen valmiin instrumentin käyttäjää. Asiakassuhde syntyy siis ohjelmoijan ja projektipäällikön välille sekä markkinoinnin ja lopullisen asiakkaan välille. Prosessijohtamisen päätarkoitus onkin organisaatioyksiköiden välisten rajaaitojen purkaminen sekä pyrkimys ulkoisten asiakassuhteiden ohella korostaa sisäistä yhteistyötä ja asiakassuhteiden merkitystä, kuten esimerkiksi tuotekehityksen ja markkinoinnin yhteistyötä. (Hammer & Stanton, 1999; Hannus, 1994; Koivula & Teikari, 1996.)

Mitä prosessiorganisaatiossa tulisi sitten johtaa? Perinteisesti johtamisosa-alueet on jaettu karkeasti ihmisiin ja asioihin. Laamanen on kuitenkin kirjassaan jakanut johtamisen osa-alueet kolmeen eri osaan: osaaminen, toimintajärjestelmät ja ihmissuhteet (Kuva 20). (Laamanen, 2001.) Saman tyyppisiä osa-alueita, mutta hiukan muunneltuna ovat esittäneet mm. Kotter (1999), Buhanist (1999) ja Buhanist et al. (2002). Esimerkiksi Buhanist et al. esittämässä EIMS mallissa management vastaa Laamasen mallin toimintajärjestelmää, osaaminen asiantuntemusta ja ihmissuhteet vuorovaikutusta. EIMS mallissa on vielä lisäksi painotettu yhtenä osa-alueena johtajan persoonaa. Tämä sisältää esimerkiksi itsenä johtamista, itsetietoisuutta, omaa visiointia, vaikuttamisen taitoa, jne. (Buhanist, 1999; Buhanist et al, 2002; Kotter, 1999.) Johtajan persoonallisuuden vaikutuksia prosessien, muutoshankkeiden ja yritysten tuloksiin on myös tutkittu erittäin paljon, kuten esimerkiksi Hughes et al (1999), Goleman et al, (2001), Sample (2002) tai Collins (2001). Sitä voitaisiin pitää myös yhtenä kehityskirjallisuuden haarana, mutta tämän osa-alueen käsittely syvällisesti tämän työn puitteissa on mahdotonta (Collins, 2001; Goleman et al, 2001; Hughes et al, 1999; Kosonen et al, 1998; Sample 2002).



Kuva 20 Organisaation johtamisen osa-alueet. (Laamanen, 2001)

6.1.1 Toimintajärjestelmä

Laamanen tarkoittaa toimintajärjestelmillä (kts. Kuva 20) kaikkia toimintatavoista tehtyjä sopimuksia, kuten organisaatiorakennetta, tavoitekirjauksia tai prosessien kuvauksia. Toimintajärjestelmä puolestaan koostuu kolmesta eri kerroksesta: ohjausmallista, yhteistyömallista ja tekemisen mallista. (Laamanen, 2001.)

Ohjausmalli

Ohjausmallilla tarkoitetaan organisaation omaksumaa toimintatapaa kuten esimerkiksi vision kirkastamista tai tavoitteiden asettamista. Ohjausmallin tarkoitus on vedota ihmisten tunteisiin niin, että ihmiset kokevat tekevänsä merkityksellistä työtä sekä ovat ylpeitä siitä ja kuulumisestaan organisaatioon. (Laamanen, 2001.) Tässä yhteydessä voitaisiin myös mielestäni puhua tietojohtamiskirjallisuudesta (Knowledge Management) tutusta käsitteestä välittäminen (Care). Tällä tarkoitetaan, että hyvällä välittämisellä ja oikealla johtamisen "mallilla" tuetun organisaation henkilöstön sisäinen luottamus kasvaa ja se alkaa toimimaan yhteistyössä kohti yhteistä päämäärää (Krogh, 1998).

Yhteistyömalli

Prosessijohtamisen tarkoituksena on parantaa organisaation sisäistä yhteistyötä (Koivula & Teikari, 1996). Tämä siksi, että organisaation tulokset eivät ole yksittäisten sankaritekojen seuraamusta, vaan yhteistyön tuloksia. Nyt siis tarvitaan toimintatapoja, jolla tätä yhteistyötä organisoidaan. Laamasen mukaan hyvällä yhteistyömallilla edistetään prosessiorganisaation tiimien ja verkostojen yhteistyötä ja pyrkimystä yhteisiin tavoitteisiin toisiinsa tukeutuen. (Laamanen, 2001.) Yhteistyömallin pitäisi myös mielestäni tukea ihmisten verkostoitumista jolloin sen kautta voitaisiin parantaa tiedon jakamista, josta esimerkiksi Hanssenin mukaan saavutetaan selkeää hyötyä

(Hansen, 1999). Mainittakoon vielä, että Vaisalan tyyppisille teknologiaintensiivisille yrityksille on ominaista tehdä prosesseja, jotka lisäävät tuotekehityksen ja myyntiosaston välistä yhteistyötä ja hallittavuutta (Glass et al, 2003).

Tekemisen malli

Tekemisen mallilla Laamanen tarkoittaa prosessin toteuttamiseksi tarvittavia työkaluja, tietojärjestelmiä, työohjeita, tekniikoita ja menetelmiä. Tekemisen mallista on kuitenkin tehtävä sopivasti löyhä, jotta malli ei rajoittaisi osaavien ihmisten työtä käytännön tilanteissa. (Laamanen, 2001; Martinsuo, 2001.)

6.1.2 Osaaminen

Toinen suuri Laamasen määrittämä osa-alue on osaamisen johtaminen, joka on hänen mielestään tärkeä osa yrityksen johtamisjärjestelmää. Myös Komulaisen mukaan sen avulla voidaan systemaattisesti ylläpitää ja kehittää strategian mukaista liiketoimintaa (Komulainen, 2001). Asiantuntijaorganisaatiossa on vielä erityisesti kiinnitettävä huomiota osaamiseen ja sen kehittämiseen, koska tunnetusti asiantuntemus on niiden pääomaa (Nahapiet & Ghoshal, 1998; Davenport, 1997). Laamanen myös toteaa, että ilman osaamista järjestelmät, prosessit ja työohjeet ovat turhia, koska ne ovat vain murto-osa siitä monimutkaisuudesta mitä niiden täytäntöönpano vaatii (Laamanen, 2001).

6.1.3 Ihmissuhteet

Kolmantena osa-alueena on ihmissuhteet. Ihmissuhteet näyttelevät tärkeää osaa koko organisaation osaamisen kehittymisessä sekä toiminnan tehokkuudessa. Huonot ihmissuhteet tai kulttuuri estävät koko organisaation tehokkaan toiminnan. (Laamanen, 2001; Krogh, 1998.) Toisaalta Schein on todennut, että organisaatiot ovat tehtäväorientoituneita ts. tiettyyn rajaan asti organisaatiot voivat toimia tehokkaastikin ilman hyviä ihmissuhteita (Schein, 1988).

6.2 Ohjelmiston suunnitteluprosessin johtaminen

Edellä on käsitelty prosessijohtamista yleisellä tasolla. Selvitetään seuraavaksi mitä yhtäläisyyksiä ja erityisvaatimuksia sillä on tuotekehitysprosessien ja erityisesti ohjelmiston suunnitteluprosessien johtamiseen. Tarkastellaan myös samalla mitä ohjelmistoprosessikirjallisuudessa sanotaan ohjelmistoprosessien johtamisesta ja johtajan roolista.

Improvisointi ja hiljaisen tiedon (tacit knowledge)[1] hyväksikäyttö ovat tunnusominaisia innovatiivisille projekteille, mutta monen tekijän koordinointi projektien aikana vaatii muodollista ja dokumentoitua prosessia (Jones, 2002). Samalla näistä yhteisistä pakollisista tukijärjestelmistä ja -rakenteista on pystyttävä tekemään mahdollisimman yksinkertaisia ja pitämään ne vähäisinä. Lisäksi niiden on oltava johdettavia ja mitattavia. Toisin sanoen tuotekehitysprosessilta vaaditaan sopivuutta todella erilaisten projektien läpiviemiseksi ja vieläpä todella erilaisilla työskentelytavoilla tehtynä. Esimerkiksi ohjelmistoprosessin tulisi istua mahdollisuuksien mukaan kaikkien

[1] tacit knowledge eli hiljainen tieto on erittäin "personaalista" ja vaikeasti mallinnettavaa tietoa. Se on myös syvästi juurtunut aktiviteetteihin, käytöntöihin, rutiineihin, arvoihin ja tunteisiin. (Nonaka et al, 2000)

ohjelmoijien henkilökohtaiseen tyyliin ja tapaan tehdä koodia. (Laamanen, 2001; Martinsuo, 2001.)

Tuotekehityksessä tekijän spesifinen osaaminen määrittää tehtävää merkittävämmin kuin tehtävän paikka prosessissa tai organisaatiossa. Tuotekehityksessä ei voi rooleja tämän takia vaihtaa samoin kuin esimerkiksi tuotannossa, jossa uuden työn oppiminen vie vähemmän aikaa. Tämä tuokin tuotekehityksen johtamiselle sekä prosesseille lisähaasteita. (Martinsuo, 2001.)

Laamanen painotti yhteisyötä ja me-hengen tärkeyttä prosessimaisessa työskentelyssä (Laamanen, 2001). Martinsuo toteaa, että tuotannon yhteishenki voi olla esimerkiksi sitoutunut yhteistä tekemistä ylläpitävään rakenteeseen tai prosessivaiheeseen. Tuotekehityksen me-henki pitäisi Martinsuon mukaan puolestaan olla organisaation näkökulmasta katsottuna tekemiseen eli prosessin läpi kulkevaan tuotteeseen sidottua. Näin pystyttäisiin ylittämään organisatorisia rajoja, jota esimerkiksi Koivula ja Teikari painottavat. (Martinsuo, 2001; Koivula & Teikari, 1996.)

Viimeisimmät ohjelmiston suunnittelumetodiikat ovat mielestäni alkaneet painottaa prosessiajattelulle tärkeitä ominaisuuksia. Niissä kiinnitetään erityistä huomiota prosessin asiakkaaseen oppimiseen, sisäiseen yhteistyöhön sekä johtajuuden uuteen rooliin.

Uusimmat agile-tyyppiset metodit painottavat erityisesti asiakkaan jatkuvaa osallistumista suunnitteluun. Ne pyrkivät täyttämään kaikki asiakkaan vaatimukset sekä vaatimusten muutokset vielä suunnittelun loppuvaiheessakin. Agile-metodit tuovat myös asiakkaan lähelle suunnittelijaa. Tällä päästään tilanteeseen, jossa esimies tai markkinointi ei enää toimi tiedon välittäjänä ja suodattimena. Nyt suunnittelija voi suoraan asiakkaan kanssa tehdä nopeasti sen hetkiseen parhaaseen tietoon perustuvia ratkaisuja. Koivula ja Teikari ovat myös tähän liittyen todenneet, että tänä päivänä asiantuntijoilla on yleensä enemmän tietoa kuin esimiehillä, joten heidän valtuuttaminen on lähes välttämätöntä nopean päätöksenteon aikaansaamiseksi. (Koivula & Teikari, 1996; Beck, 2000; Highsmith, 1999; Grenning, 2001; Teikari, 2000.)

Nykypäiväinen asiantuntija- ja prosessijohtajuus vaatii enemmän valmentajan roolin ottamista. Tätä mieltä ovat esimerkiksi Hammer & Stanton, Laamanen ja Koivula & Teikari. Mielestäni voitaisiin puhua myös siirtymisestä puhtaasti management-tyylisestä johtamisesta leadership-mangement-yhdistelmään (Hughes et al, 1999; Buhanist, 1999). Tämä on selvästi tiedostettu myös agile-metodien kehittäjien taholta. Esimerkiksi valmentajuuden (coach) ja leadership-roolin ottamista korostavat erityisesti Beck (XP) ja Highsmith (ASD). (Beck, 2000; Hammer & Stanton, 1999; Highsmith, 1999; Koivula & Teikari, 1996; Laamanen, 2001.)

Oppiminen on yksi prosessiajattelun kulmakivistä (Koivula & Teikari, 1996; Laamanen, 2001). On siis pystyttävä oppimaan virheistään ja tekemisistään. Tavallinen vesiputousmallinen prosessi ei tätä kovinkaan hyvin tue. Sitä vastoin agile-metodeihin oppiminen on rakennettu toistuvaksi ja automaattiseksi toiminnaksi. Tämän on mahdollistanut iteratiivinen suunnittelu. Kun iteratiivisten jaksojen pituutta vielä agile-oppien mukaan lyhennetään voidaan saada aikaan tilanne, jossa oppiminen saadaan lähes jatkuvaksi. (Abrahamsson et al, 2002; Beck, 2000; Highsmith, 1999; Malatouxin, 2003; Gilb, 1997.)

Kuten Teikarikin lehtihaastattelussa toteaa, on erakkomainen yksinään huoneessaan suunnittelu asiantuntijatyössä väistymässä (Teikari, 1999). Tähän ovat heränneet myös agile-metodit ja sisäinen yhteistyö ja kehittäminen on tullutkin yhdeksi niiden manifestin neljästä kohdasta[1]. Useiden metodien ohjelmointitavat on suunniteltu monen ohjelmoijan yhteistyötä tukevaksi tai "pakottavaksi". Esimerkiksi XP perustuu täysin kahden ohjelmoijan yhteistyöhön eli ns. pariohjelmointiin. Sisäistä yhteistyötä vielä korostetaan fyysisillä tilaratkaisuilla laittamalla ohjelmoijat istumaan fyysisesti samaan tilaan sekä vierekkäin. Sijoittelulle ja kommunikointitavoille annetaan jopa suhteellisen tarkkoja ohjeita. (Beck, 2000.)

Ohjelmistoprosessikirjallisuudesta löytyy siis paljon yhtäläisyyksiä perinteiseen ja tunnetumpaan "tavallisen" yrityksen prosessijohtamiseen. Osaksi tämä varmasti johtuu siitä, että ohjelmistoprosessikirjat ovat pääosin suunnattu johtajille ja niiden kirjoittajat ovat yleensä toimineet suurten tai vaativien projektin vetäjinä sekä yrityskonsultteina. Esimerkkinä tästä voisi mainita ASD:n kehittäjän Highsmith, joka on toiminut Apollo-avaruusalusohjelmassa sekä kolmekymmentä vuotta projektipäällikkönä, konsulttina ja kirjoittajana (Highsmith, 1999).

6.3 Prosessimittarit

Prosessimittarit ovat yksi tärkeimmistä johdon työvälineistä hallita prosesseja sekä seurata niiden tehokkuutta. Ilman prosessimittareita ei voitaisi edes tietää onko laatutavoitteet tai prosessin tehokkuustavoitteet täyttyneet. Prosessimittarit ovat myös kehityshankkeen kannalta tärkeitä, koska niiden avulla voidaan kehityshankkeen tuotosta arvioida sekä todistaa sen tuoma hyöty (Kosonen et al, 1998).

Prosessimittarit voidaan jakaa karkeasti kahteen tyyppiin: prosessin aikaisiin (in-process) ja jälkeisiin (post-process) mittareihin (Cooper, 2000). Lisäksi voitaiin puhua myös kovista ja pehmeistä mittareista (Kosonen et al, 1998). Prosessin suorituskykyä voidaan mitata käyttäen eri teemoja, kuten aikaa, rahaa, määrää, fysikaalisia ominaisuuksia tai sidosryhmien näkemyksiä (Laamanen, 2001). Valittavat mittarit tulisi rakentaa sellaisiksi, että ne ovat selkeitä, eivät manipuloitavissa ja että ne mittaavat tärkeitä sekä oleellisia asioita. Lisäksi niiden määrä on syytä pitää suhteellisen vähäisenä. Mittarin käyttö ei myöskään saisi olla kallista eikä viedä liikaa aikaa. (Lecklin, 2002.) Jos kehitetään uusia mittareita, on luonnollista, että mittarin on oltava sellainen, jolla pystytään mittaroimaan myös menneitä projekteja. Muussa tapauksessa mitataan vain tulevaa eikä tiedetä miten prosessin parannus auttoi vai auttoiko lainkaan.

Ohjelmistoprosesseja voidaan mitata tavallisilla prosessimittareilla kuten esimerkiksi läpimenoajalla tai tehokkuudella. Tämän lisäksi on olemassa joukko mittareita, jotka on suunniteltu juuri ohjelmiston suunnittelun tehokkuuden ja laadun tarkkailun mittarointia varten. Näitä ovat esimerkiksi Line of Code (LOC), Halsted and cost per defect, Complexity metrics ja Function points mittarit. Näistä kaksi ensimmäistä ovat saaneet kritiikkiä esimerkiksi siitä, että ne ovat riippuvaisia ohjelmointikielestä. Lisäksi Jones mainitsee, että luotettavien ja oikeita asioita mittaavien mittareiden tekeminen "hybridi-tuotteille", jotka sisältävät ohjelmistoa ja elektroniikkaa on erittäin vaikeaa tai jopa mahdotonta. (Jones, 1994.)

[1] http://www.agilemanifesto.org/

Kehitettävään prosessiin tullaan tekemään ainakin yksi ohjelmiston laadullinen mittari. Tämä esitellään tämän työn kehitysvaiheessa.

6.4 Prosessi vallan välineenä

Pohditaan lopuksi prosessien käyttöä vallan välineenä. Ideaalinen prosessihan olisi sellainen, jonka jokaista komponenttia voitaisiin tarvittaessa mittaroida ja ohjata. Prosessi olisi tällöin täysin hallinnassa ja sen tuotos olisi aina haluttu. Tähän selvästi prosessikirjallisuus pyrkii. Tosin asiantuntijoille tarkoitetuista prosesseista on sanottu, että niiden pitäisi olla löyhiä, ja johtaminen pitäisi olla valmentajamaista (Koivula & Teikari, 1996; Martinsuo, 2001; Laamanen, 2001).

Täydellinen prosessi mielestäni johtaisi Focaultilaisen vallan näkökulman kautta tilanteeseen, joka olisi täydellinen "panopticon": Prosessi tekisi työpaikasta paikan jossa työntekijä tietää, että hänen työtään voidaan valvoa hetkenä minä hyvänsä, mutta tätä hetkeä hän ei taas tiedä. Tämän takia työntekijän itsekontrolli kasvaisi ja hän alkaisi käyttäytyä niin kuin hän olisi kokoajan valvonnan alla. Toisin sanoen täydellinen prosessi käyttää tehokkaasti biovaltaa työntekijää kohtaan kurinpidollisen vallan lisäksi. Prosessikirjallisuus mielestäni keskittyykin enemmän kurinpidollisen vallan hallintaan ja lisäämiseen, kuten esimerkiksi prosessin katselmuksien ja mittareiden suunnitteluun. (Buchanan & Badham, 1999.) Tämä on mielestäni mielenkiintoinen asia, mutta ovatkohan prosessikirjallisuuden tekijätkin havainneet saman asian?

7 KEHITYKSEN KULKU

Tässä luvussa esitellään kehityshankkeen aikana tehdyt toimenpiteet ja niistä saadut tulokset. Kehityksen kulkua voi verrata kuvan 2 prosessikuvaukseen seuraamalla tämä luvun otsikoiden suluissa olevia nimiä. Tässä luvussa pyritään kertomaan vain tehdyt toimenpiteet ja välttämään toimenpiteiden tai saatujen tulosten liiallista analysointia, koska niiden analysointiin ja pohdintaan on keskitytty luvuissa 8 ja 9 .

7.1 Kehityshankkeen aloitus

Kehityshanke sai alkunsa tarpeesta kehittää paremmin hallittava, kuvattu sekä dokumentoitu sulautetun ohjelmiston suunnitteluprosessi. Tarpeen todellista alkulähdettä on vaikeata määritellä, mutta visio tavoitteesta jalostui tuotekehityksen prosessinomistajan, tuotekehityspäälliköiden ja ohjelmiston kehitysryhmän vetäjän kesken. Yhtenä työkaluna tässä käytettiin sulautetun ohjelmiston suunnittelijoiden kompetenssien määrittelyä.

7.1.1 Kehitysstrategia (Road Map)

Vuotuisten strategiasuunnittelujen tuloksena ohjelmiston kehitysryhmän vetäjä, pienellä avustuksellani, kokosi kaikki sulautetun ohjelmiston tuotantoon liittyvät kehityshankkeet tiekartaksi (road map) eli loogiseksi kehityshankkeiden kokonaisuudeksi. Tässä Road Mapissa oli yhtenä janana suunnitteluprosessi ja sen kehittäminen. Kehityskohteiden sijoittelussa otettiin huomioon tulevat tuoteprojektit, henkilöstöresurssit sekä muut hankkeisiin vaikuttavat tekijät.

Road Map hyväksyttiin prosessin omistajan toimesta strategiakokouksessa. Ohjelmiston suunnitteluprosessin kehityshankeen aloitusajankohta oli määritetty vuoden 2003 helmikuun paikkeille. Suunniteltu kehityshanke päätettiin toteuttaa Vaisalan kehitysmallin mukaisesti (Kuva 2), ja sen ensimmäinen vaihe oli kehityskatselmuksen (Development Review 0, DVR0) valmistelu ja järjestäminen.

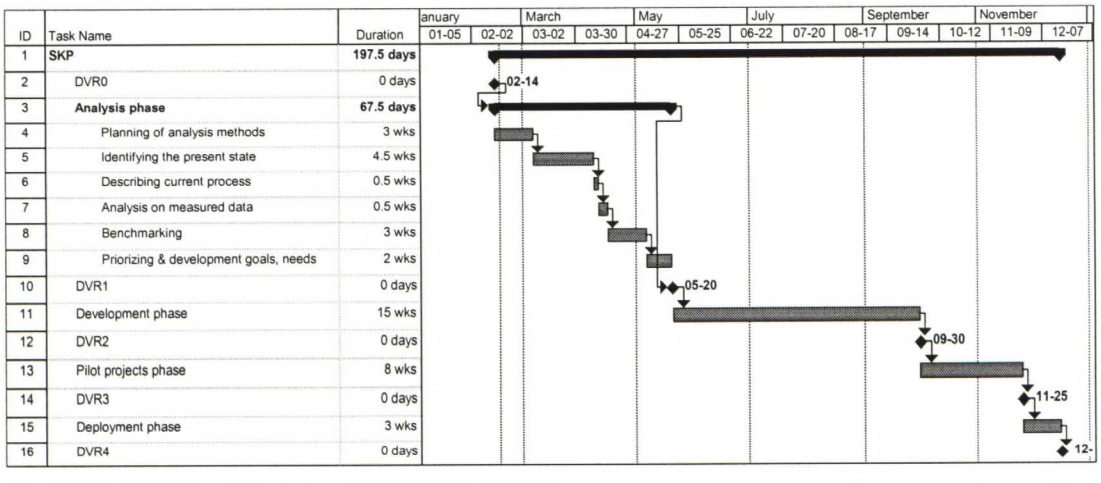
7.1.2 Kehityskatselmus 0 (DVR0)

Kehityskatselmus ja samalla hankeen aloituspalaveri pidettiin 14.2.2003. Katselmukseen osallistuivat ohjausryhmä sekä kehitysryhmä. DVR0 tilaisuuden päätarkoituksena oli hyväksyä kehittämishankkeen aikataulu ja sisältö. Samalla tilaisuuden tarkoituksena oli määrittää hankeen resurssit, rajaus ja tarkistaa projektisuunnitelma. Hankkeen projektipäällikkö Tuominen esitteli tilaisuudessa ohjausryhmälle projektisuunnitelman, jonka sisältämiä asioita on listattu taulukossa 2.

Taulukko 2 Projektisuunnitelman sisältämiä kohtia.

- Projektin tausta, sisältö ja aikataulu
- Projektin resurssit ja budjetti
- Kommunikointisuunnitelma
- Dokumentointisuunnitelma
- Projektin laadun varmennussuunnitelma
- Suunnitelma analyysivaiheen toteuttamisesta
- Alustava kuvaus koko hankkeen aikataulusta

Ohjausryhmä hyväksyi Tuomisen esityksen ja sen pohjalta lähdettiin liikkeelle. Kuvassa 21 on projektin alustava aikataulu. Analyysivaihe kestää projektiaikataulun mukaan noin kolmisen kuukautta ja se alkaa nykytilan arvioinnilla. Projektiaikataulu on suunnittelu ja jalkauttamisvaiheiden kohdilta ylimalkainen, koska niiden työvaiheet sekä tarkemmat aikataulut tullaan tarkentamaan hankeen edetessä.



Kuva 21 SKP:n alustava projektiaikataulu.

Katselmuksessa päätettiin myös ottaa kehityshankkeen ohjausryhmään mukaan kolmas henkilö. Valittu henkilö oli tehnyt edellisessä yrityksessään vastaavanlaista kehitystyötä, jonka toivottiin nyt osaltaan auttavan tätä hanketta. Ehdotus kolmannesta henkilöstä tuli prosessin omistajalta.

7.2 Analyysivaihe (Analysis)

Analyysivaiheessa pyritään tuottamaan yhteinen käsitys nykyisestä sulautetun ohjelmiston suunnittelun tilasta sekä kirkastamaan ohjausryhmän sekä erityisesti kehitysryhmän visiota. Samalla viestitään organisaation muille jäsenille tulevasta muutoksesta sekä kerätään tuloksia myöhemmin tapahtuvaa kehityshankkeen arviointia varten. Edellä esitettyjen asioiden saavuttamiseksi on kehitysprosessissa määritetty toimenpiteitä jotka pitää suorittaa (Taulukko 3) sekä asioita jotka kehityssuunnitelmasta pitää löytyä (Taulukko 4).

Taulukko 3 Toimenpiteet jotka vähintään tulee suorittaa analyysivaiheessa.

• Asiakas-/käyttäjävaatimusten ja kehitystarpeiden määrittäminen
• Nykytilan sekä sen suorituskyvyn määrittäminen
• Nykyisen prosessin kuvauksen tekeminen
• Vahvuuksien, heikkouksien ja mahdollisuuksien analysoiminen
• Sopivan kehityksen mittaustavan tarjoaminen
• Gap-analyysin muodostaminen
• Analyysin pohjalta kehityskohteiden priorisoiminen
• Projektin konkreettisten tavoitteiden määrittäminen

Taulukko 4 Asiat jotka pitää täydentää projekti- / kehityssuunnitelmaan analyysivaiheessa.

• Projektin tavoitteet ja metodit kuinka niitä mitataan
• Yksityiskohtainen toimintasuunnitelma
• Lista (ja alustava toimintasuunnitelma, jos mahdollista) loppuosalle kehityshanketta
• Hahmotelma pilotti-, kehittämis- sekä jatkuvan kehittämisen vaiheista
• Kommunikointisuunnitelma kehitysvaiheeseen

Taulukoiden 3 ja 4 vaatimuksien täyttämiseksi suunniteltiin yksityiskohtainen työvaihesuunnitelma. Työvaiheiden hahmottelu tapahtui Tuomisen ja minun toimesta ja lopuksi niistä keskusteltiin yhteisesti kehitysryhmän kanssa. Yhteisillä päätöksillä päätettiin nykytilan analysointivaiheen toteuttamiseksi järjestää nykytilan arviointitilaisuus sekä benchmarking-vierailuja. Seuraavien kappaleiden tarkoituksena on kuvata nämä sekä muut analyysivaiheen työvaiheet.

7.2.1 Nykytilan arvioinnin suunnittelu

Nykytilan arvioinnin suunnittelu oli ensimmäinen kehitysryhmän yhteinen tehtävä. Tästä voisi katsoa alkaneen myös ryhmän yhteistyö ja roolien muodostuminen. Ryhmän toimintatapa perustui viikoittaisiin työpalavereihin. Työpalavereihin saatettiin myös kutsua ohjausryhmä mukaan, jos tämä nähtiin tarpeelliseksi.

Koska nykyisestä sulautetun ohjelmiston suunnittelusta ei ollut olemassa mitään dokumentoitua prosessia, kulki ohjelmiston suunnittelu nykyisen tuoteprosessin mukana. Tämän takia ensimmäisenä tehtävänä oli suunnitella menetelmä, jolla nykyinen suunnittelukäytäntö saataisiin mallinnettua. Mallintamisen apuna käytettiin hyväksi prosessimääritelmiä ja ajattelutapaa, joihin luvussa 3 on viitattu.

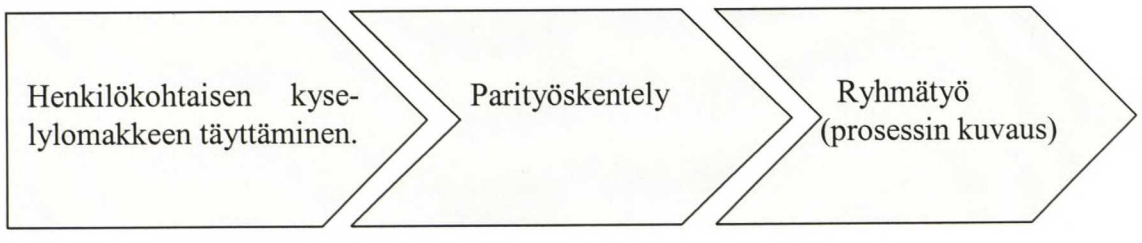
Pyrinkin ihan aluksi selvittämään eri tiedonlähteistä kuinka kuvaamattomia menetelmiä yleensä yritetään mallintaa. Mitään yksiselitteistä ohjetta tai työtapaa en pys-

tynyt löytämään. Professori Veikko Teikarilta sain mallintamisen ohjeeksi, että "Käytä tervettä järkeä" (Teikari, 2003). Tämän ohjeen lisäksi Scheinin kirjasta minulle oli jäänyt mieleen eräs lausahdus: "Kysyminen henkilöltä yhteisistä ilmiöistä (prosesseista) on tehotonta ja mahdollisesti pätemätöntä" (Schein, 2001). Toisin sanoen tässä tapauksessa ryhmähaastattelu olisi paras vaihtoehto Scheinin mukaan.

Tämän esiselvitysvaiheen pohjalta ja eri diagnoosivaihtoehdot kehitysryhmälle esiteltyäni päätettiin ryhmän kanssa yhdessä valita analysointitavaksi nykytrendin mukaisesti workshop-tyylinen tilaisuus. Tilaisuus suunniteltiin järjestäväksi ohjelmistointressiryhmän saunaillan yhteydessä, johon kutsuttiin kaikki Vaisala Instrumentsin sulautettujen ohjelmistojen suunnittelijat.

7.2.2 Workshopin sisältö

Workshopin sisällön suunnittelu aloitettiin siten, että tein ensin kehitysryhmälle luonnoksen sisällöstä. Luonnoksen pohjalta jalostettiin ryhmän kanssa lopullinen ratkaisu Workshopin aikatauluksi ja sisällöksi. Workshopin esittelykalvot ovat liitteenä (Liite 1, Workshopin aikataulu ja sisältö).



Kuva 22 Karkea kuvaus workshopin sisällöstä ja kulusta.

Workshop sisälsi kolme eri vaihetta, jotka on esitelty kuvassa 22. Ensimmäinen vaihe oli henkilökohtainen työskentely, jonka tarkoituksena oli pistää osallistujat miettimään kyselylomakkeen (Liite 2, Henkilökohtaiset kysymykset) avulla mitä he oikein tekevät. Kysymykset oli laadittu siten, että ne pyrkivät auttamaan vastaajaa hahmottamaan koko ohjelmiston suunnittelun kokonaisuuden. Mainittakoon tässä kohtaa, että sana prosessi oli suunnittelijoiden keskuudessa lähes kirosana, joten kysymyksissä ja tilaisuudessa pyrittiin välttämään sen ja muiden "erikoissanojen" käyttöä.

Tarkoituksena oli toteuttaa tuplatiimiajatusta. Perusajatuksena siinä oli pistää ihmiset ensin miettimään itsenäisesti, tämän jälkeen keskustelemaan pareittain ja vasta lopuksi keskustelemaan isommassa ryhmässä. (Laamanen, 2001.) Parityöskentelyn pohjana käytettiin kyselylomaketta. Ryhmätyössä puolestaan käytettiin hyväksi parityöskentelyn tuotoksena jalostettuja vastauksia ja ideoita. Tilaisuuden päätteeksi tehtävän ryhmätyön tarkoituksena oli kuvata graafisesti nykyinen suunnitteluprosessi edellisten työvaiheiden avulla.

Workshopin puhemiehenä toimi kehityshankkeen vetäjä Tuominen. Muiden kehitysryhmään kuuluvien henkilöiden tehtävänä oli havainnointi, tiedonkerääminen ja tilaisuuden käytännön järjestelyistä vastaaminen.

7.2.3 Workshopin tulokset

Workshopiin osallistujat ja kehitysryhmä pitivät workshopia tarpeellisena sekä onnistuneena. Workshopin aikataulu piti hyvin paikkansa ja kaikki suunnitellut vaiheet saatiin käytyä läpi. Tilaisuuden agendaan tehtiin myös viimehetken muutos. Muutosehdotus tuli yhdeltä ohjausryhmän jäseneltä. Hänen ehdotuksensa oli, että tilaisuuden yhteydessä kysyttäisiin suunnittelijoilta vielä, mikä olisi heidän mielestään hyvä tavoitetila. Halusimme kuitenkin pitää visusti erillään nykytilan ja tavoitetilan, koska muuten nykytilasta kysyttäessä olisi helposti vaarana nyky- ja tavoitetilan sekoittuminen. Tämän takia nykytilasta keskusteltiin aivan viimeiseksi.

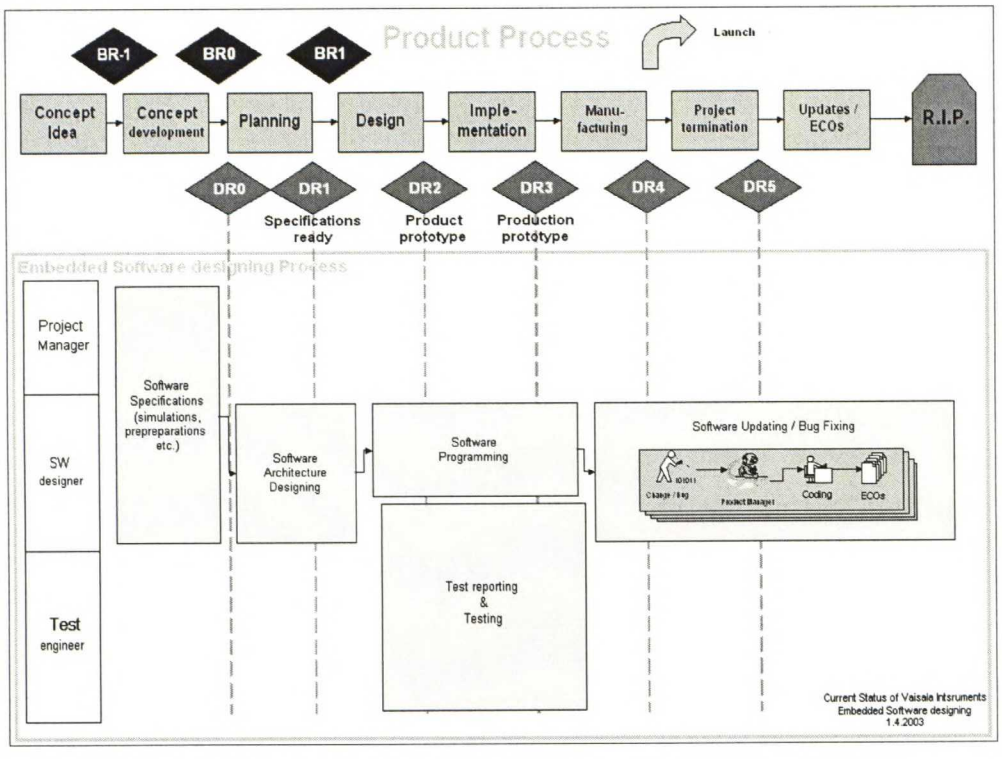
Materiaalinen anti Workshopista oli kyselylomakkeet sekä neljä kappaletta ryhmien piirtämiä prosessikaavioita. Prosessikaaviot ovat puhtaaksi piirrettyinä liitteinä (Liite 3, Ryhmän 1 tekemä nykytilan prosessikuvaus ja Liite 4, Ryhmän 2 tekemä nykytilan prosessikuvaus). Tavoitetilan prosessikaavioita emme piirtäneet puhtaaksi, mutta niitä käytettiin sellaisenaan kehityspalavereissa apuna.

7.2.4 Workshopin tulosten yhteenveto

Ryhmien tekemät prosessikuvaukset ja kyselylomakkeet vedettiin kehitysryhmän toimesta yhteen ja niiden pohjalta hahmoteltiin työnkulusta sekä prosessin dokumenteista erilliset kuvaukset. Tämän tarkoituksena oli selkeyttää ohjelmiston suunnittelun kulkua ja auttaa kokonaisuuden ymmärtämistä.

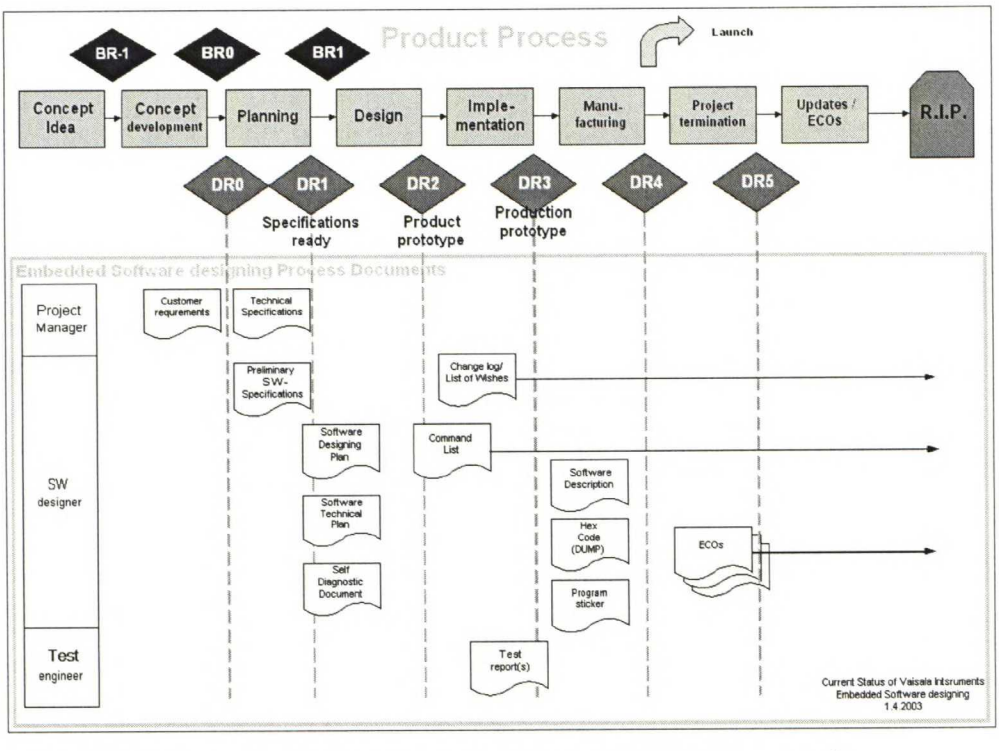
Eri tehtäviä ohjelmistosuunnittelusta löydettiin useita. Toisaalta oli myös mahdotonta eritellä joitakin tehtäviä toisistaan. Esimerkiksi käyttöliittymän ja mittauksen suunnittelun erottelu ohjelmointivaiheessa ei ollut mielekästä, koska ei voida yksiselitteisesti sanoa kumpaa suunnittelija tekee ohjelmoidessaan.

Kuva 23 kuvaa nykyisiä sulautetun ohjelmiston suunnitteluprosessin työvaiheita. Kuvauksessa on otettu käyttöön esimerkiksi Laamasen kirjassa esitetty työvaiheen tekijöiden määrittäminen. Työnvaiheiden tekijät ovat määritelty kaavion vasemmassa laidassa. (Laamanen, 2001.)



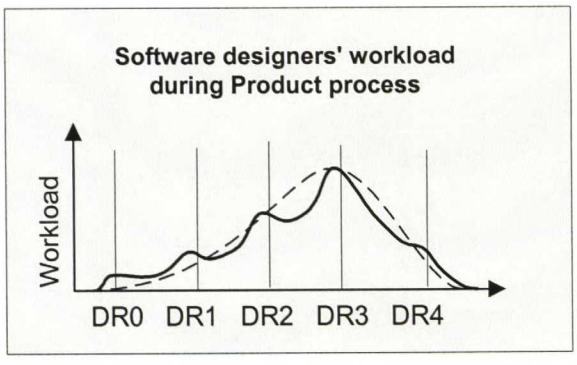
Kuva 23 Ohjelmistosuunnittelun työnkulun kuvaus ryhmätöiden pohjalta.

Suunnittelun aikana tehtyjä dokumentteja määrittäessä käytettiin avuksi kyselyn tuloksia. Vastauksista laskettiin tehdyt dokumentit ja dokumenttien painoarvoa peilattiin suunnittelijan kokemukseen. Sulautetun ohjelmiston suunnitteluprosessin aikana syntyvät dokumentit on esitetty prosessikuvauksessa kuvassa 24.



Kuva 24 Ohjelmistosuunnittelu prosessin aikana syntyvät dokumentit.

Yksi mielenkiintoinen asia tuli esiin Workshopissa ryhmäkeskustelujen aikana. Nimittäin suunnittelijat kokivat nykyisen prosessin kuormituksen kasvavan loppua kohden. Tätä tarkemmin analysoitaessa havaittiin, että suurin kuormituspiikki oli DR3-katselmuksen kohdalla. Kuormitusta havainnollistettiin workshopissa hahmotellun kuvan 25 mukaisella käyrällä.



Kuva 25 Ohjelmistosuunnittelijan kuormittuneisuus tuoteprosessin aikana.

7.2.5 Nykytilan hyvät ja huonot puolet

Workshopin tulokset esiteltiin myös kaikille ohjelmiston suunnittelijoille kuukausittain järjestettävissä yhteisessä palaverissa. Kaikki ohjelmistosuunnittelijat hyväksyivät kuvatun prosessin (Kuvat 23 ja 24). Näin oli saatu yhteisesti hyväksytty käsitys nykyisestä sulautetun ohjelmiston suunnittelun kulusta.

Seuraavana tehtävänä kehitysryhmällä oli etsiä nykyisestä käytännöstä hyvät sekä huonot puolet. Taulukossa 5 on listattu muutamia havaintoja, joita nykykäytännön ja tavoitetilan väliltä löydettiin.

Taulukko 5 Nykykäytännön ja tavoitetilan eroja.

Hyviä puolia:
- ✓ Joustava
- ✓ Suunnittelija saa toteuttaa itseään

Huonoja puolia:
- Ei määritelty
- Ohjelmoija voi toteuttaa itseään
- Katselmoinnit puuttuvat
- Ohjelmointi ei johdettavissa / "seurattavissa"
- Dokumentointi puutteellista
- Ohjelmoija testaa itse oman koodinsa
- Nykyinen tuoteprosessi ei tue ohjelmistosuunnittelua
- Ei selkeitä ohjelmointivaiheita (ei välttämättä huono asia)
- Ei versionhallintaa
- Ylläpito joskus hankalaa, kun ei sovittua dokumentointitapaa
- Ohjelmiston koodin ja testauksen jäljitettävyys heikkoa
- Koodista ei kuvausta, vain koodissa itsessään olevia vähäisiä kommentteja
- Testausta ei suunnitella
- Valmiista ohjelmasta ei kuvausta

Listasta tuli aika negatiivinen, mutta se oli odotettavaa, koska tarkoituksenahan oli loppujen lopuksi kyseenalaistaa nykykäytäntöä. Toisaalta nykykäytännöstä ei ollut paljon hyvää sanottavaa prosessiajattelun näkökulmasta. Tässä vaiheessa kehityshanketta oli selvästi nähtävissä, että myös kaikki ohjelmiston suunnittelijat kyseenalaistivat nykyisen suunnittelukäytännön.

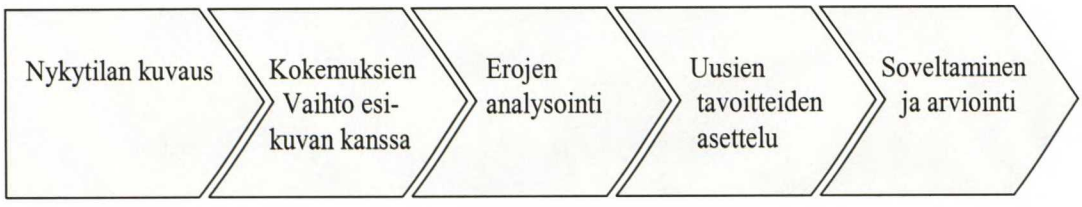
7.2.6 Benchmarking

Seuraava suuri toiminnallinen askel nykytilan arviosta oli benchmarking. Benchmarkingin tarkoituksena oli tutkia mitä eri sulautetun ohjelmiston suunnittelukäytäntöjä oli muissa yrityksissä käytössä. Benchmarkingin toivottiin tuovan uusia ideoita ja ajatuksia prosessin kehittämiseen.

Benchmarkingista, kuten arvata saattaa, löytyy kirjallisuudesta vaikka kuinka paljon aineistoa. Benchmarking ei ole kuitenkaan tämän työn aihe, joten sen käytännöistä ja teorioista kerrotaan vain pintapuolisesti. Benchmarking on kuitenkin mielestäni erittäin tärkeä osa prosessien kehittämistä ja miksei sitä voisi käyttää myös hyväksi muidenkin organisaation kehitysosa-alueiden kehityksen tukena.

Benchmarkingin suunnittelu

Benchmarkingissa noudatettiin Laatukeskuksen benchmarking-mallin kuvan 26 kolmea ensimmäistä vaihetta. Kaksi viimeistä vaihetta ottavat kantaa prosessin kehittämiseen, jotka suoritimme oman kehitysprosessin mukaisesti. (Laatukeskus, 1998.)



Kuva 26 Benchmarking-malli (Laatukeskus, 1998).

Nykytilan kuvaaminen oli benchmarking-mallin ensimmäinen vaihe, joka oli tässä kehityshankkeessa tehty workshopin avulla. Seuraavana vaiheena oli kokemuksien vaihto. Ennen kokemusten vaihtoa suoritettiin taulukossa 6 listatut toimenpiteet:

Taulukko 6 Benchmarkingin esivalmistelutoimenpiteitä

• Yhteydenotot kiinnostaviin yrityksiin
• Benchmarking-tilaisuuden ajankohdan ja sääntöjen sopiminen
• Benchmarkattavien asioiden selvittäminen
• Oman yrityksen, nykyisen prosessin ja alustavien kysymysten vaihto benchmarking-kohteen kanssa.
• Muut käytännön järjestelyt

Benchmarkingin tulokset

Muiden yritysten suunnittelukäytäntöihin tutustuminen tapahtui viikolla 20. Benchmarking-kumppaneiksi oli valittu kaksi kotimaista elektroniikka-alan yritystä, jotka eivät ole Vaisalan kilpailijoita. Suhteiden luomisessa käytettiin hyväksi henkilökohtaisia sekä vanhoja benchmarking-verkostoja.

Benchmarking vierailujen jälkeen koostettiin kummastakin kohteesta arviointidokumentit. Arvioinnissa yritettiin löytää kohteiden prosessien hyviä ja huonoja puolia. Dokumentit lähetettiin myös benchmarking-kumppanille, jos he niin halusivat. Näin he saivat myös arvokasta tietoa ulkopuolisen näkökulmasta.

Benchmarking-kohteiden prosessikuvauksia ei ole tässä työssä syytä lähteä kovin tarkasti esittelemään. Seuraavaksi kuitenkin kerron lyhyenä yhteenvetona kummankin kohteen prosessien toimintatavat.

Ensimmäisestä kohteesta on aluksi hyvä mainita se, että se oli noin kaksi vuotta sitten aivan samassa tilanteessa kuin missä meidän kehitysryhmämme nyt on. Tämä kohde oli ratkaissut oman ongelmansa evoluutiomaisella iteratiivisella prosessilla, joka pohjautui heidän omaan kokemukseensa sekä Niels Malatouxin Evolutionary Project Management Methods eli Evo-menetelmään (Malatouxin, 2003). Heidän prosessinsa oli erityisesti räätälöity Evoa mukaillen sulautetun ohjelmiston ylläpitoa varten. Tämän takia se ei ollut suoraan sovellettavissa meidän käyttöömme, koska

tarvitsemme ohjelmistotuotteen alusta loppuun kattavaa ohjelmistoprosessia. Päivän kestänyt benchmarking antoi kuitenkin paljon hyviä ideoita sekä ajatuksia siitä, kuinka heidän prosessiaan voisi käyttää hyväksi myös koko ohjelmiston elämänkaaren kattavassa ohjelmiston suunnitteluprosessissa.

Yhtenä erikoishuomiona voisi ensimmäisen kohteen prosessista esitellä prosessin yksittäiselle suunnittelijalle aiheuttaman kuormituksen jakautumisen prosessin aikana. Ainakin Malatouxin mukaan suunnittelijoiden kuormituksen pitäisi jakautua kuvan 27 tavalla (vrt. kuva 25). Uskon, että tämä toteutui tässä benchmarking-kohteessa.

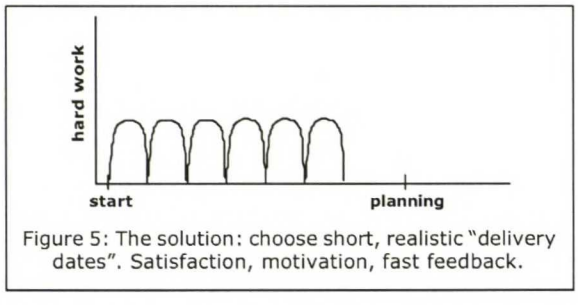


Figure 5: The solution: choose short, realistic "delivery dates". Satisfaction, motivation, fast feedback.

Kuva 27 Ohjelmistosuunnittelijan kuormittuneisuus Evolutionary prosessia käytettäessä. (Malatouxin, 2003)

Toinen kohde oli yrityksenä hiukan perinteikkäämpi. Tässä yrityksessä käytettiin niin sanottuun vesiputousmalliin perustuvaa ohjelmiston kehitysprosessia. Prosessi oli heillä erittäin tarkasti kuvattu ja sitä voisi kutsua byrokraattiseksi. Prosessi pyrki dokumentoimaan kaiken mahdollisen ja näin siitä oli saatu myös erittäin raskas. Se perustui Jaaksi et al. kirjaan Tried & True Object Oriented Development (Jaaksi et al, 1999). Prosessi oli suunniteltu ensisijaisesti objektiorientoituneeseen ohjelmointiin, joten se ei suoraan sovi meidän suunnittelutapaamme. Prosessista voi kuitenkin nähdä sen tärkeimmän ajatuksen, eli vesiputousmallin ja sen kuinka sitä voidaan käyttää ohjelmistojen suunnittelussa.

7.2.7 Prosessimittarien kehittäminen

Jotta suunnitellun prosessin toimivuutta voitaisiin arvioida myöhemmin piti sille kehittää prosessimittareita. Vaisalan kehitysprosessi vaatii myös mittareiden kehittämisen aloittamisen tässä kehityshankkeen vaiheessa. Kuten jo teoriaosuudessa mainittu, on ohjelmistoteollisuuden prosessimittarit oma tieteen alansa ja niiden sellaisenaan hyödyntäminen juuri Vaisalan Instrumentsin tapaukseen on vaikeaa. Tämän takia kehitysryhmä päätti kehittää oman mittarin, joka soveltuu tuoteprosessin aliprosessina toimivan sulautetun ohjelmiston suunnitteluprosessin mittaamiseen. Tämä eroaa tavallisista ohjelmistotuotannon mittareista esimerkiksi siten, että se ei mittaa prosessin taloudellisia suureita, koska ne ovat mekaniikan, elektroniikan ja ohjelmiston yhteistuloksia. Kehitetyn mittarin päätarkoitus onkin mitata aliprosessin laatua.

Yhdeksi mittariksi ehdotettiin "Bug-ECO" mittaria, joka mittaisi ohjelmistoon jääneiden virheiden määrän sen tuotantoon luovuttamisen jälkeen. Tämä myös kirjallisuudesta tuttu ja yksinkertaiselta kuulostava mittari käytännössä hylättiin jo ennen

DVR1-tilaisuutta kehitysryhmän omasta toimesta, koska mittarin vaatiman tiedon kerääminen olisi erittäin vaikeata sekä tulkinnanvaraista.

Varteenotettavimmaksi mittariksi nousi lopulta "Source and hex code ratio" nimellä kutsuttu mittari. Tämän mittarin tarkoitus on mitata ohjelmiston lähdekoodin kommentoinnin määrää ja laatua. Tämä tapahtuu vertaamalla lähdekoodin kokoa käännettyyn heksatiedostoon. Kun lähdekoodi vielä pakataan ennen vertailun tapahtumista, pystytään poistamaan toistuvat kommentoinnit, eli ns. copy-paste kommentointi ei auta. Tämä mittari on siinä mielessä hyvä, että se mittaa oikeaa asiaa eli koodin luettavuutta, lisäksi se on helppo käyttää ja mittarilla voidaan laskea helposti myös taaksepäin. Lisäksi se tukee esimerkiksi XP-tyyppistä agile-metodia, joka korostaa lähdekoodin kommentointia (Beck, 2000).

Kehitetyt mittarit esiteltiin ohjausryhmälle esimerkkinä mittareista DVR1 tilaisuudessa. Lisäksi tilaisuutta ennen oli "Source and hex code ratio" mittarin mukaisesti laskettu muutamia esimerkkitapauksia, mittarin toiminnan havainnollistamiseksi.

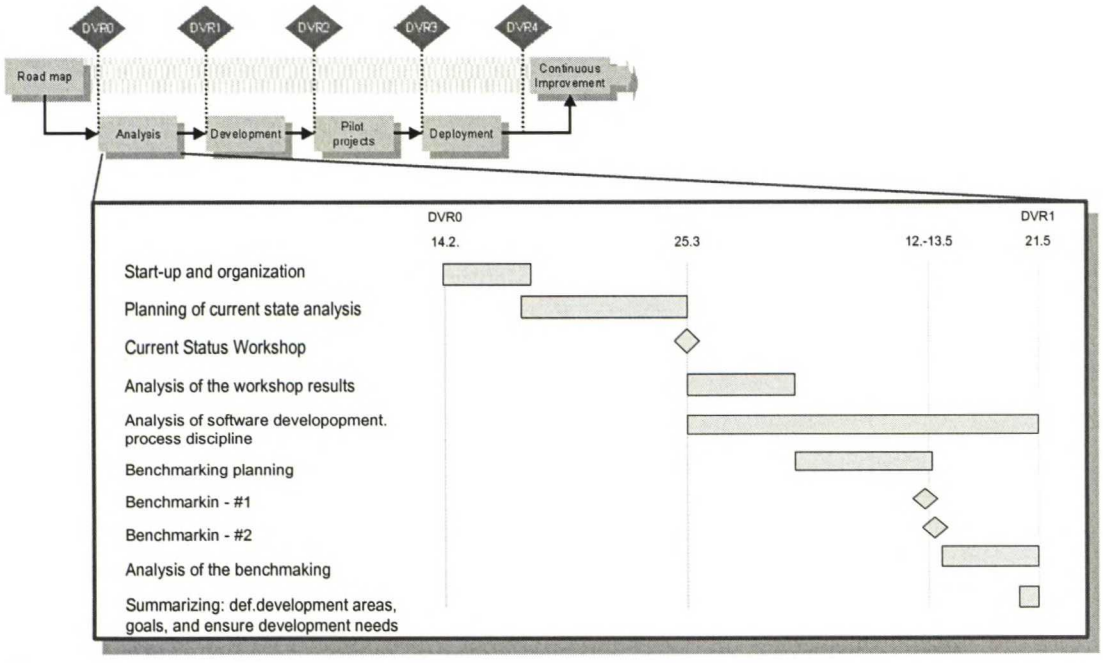
7.2.8 Kehityskatselmus 1 (DVR1)

Analyysivaihe päättyi muodollisesti DVR1-tilaisuuteen (kts. Kuva 2). DVR1 tilaisuuden tarkoituksena oli katselmoida analyysivaiheen tulokset sekä tehdyt toimenpiteet. Näiden pohjalta oli tarkoitus tarkentaa kehitysvaiheen aikataulua sekä parantaa kehitysprosessin mukaan projektisuunnitelmaa taulukossa 7 listatuin osin.

Taulukko 7 DVR1-tilaisuuteen tarkistettavat asiat.

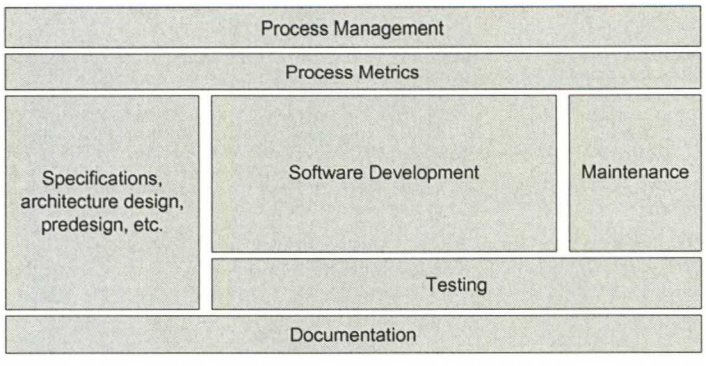
• Ehdotus kehityskohteista a) kehitysprosessin aikana b) ja jatkuvan kehityksen aikana
• Kehitysvaiheen toimintasuunnitelma ja aikataulu
• Kehitysvaiheen resurssit ja vastuukysymykset
• Pilottivaiheen alustava suunnitelma

Kehityskatselmuksessa esiteltiin ohjausryhmälle kuvan 28 mukainen aikajana toteutetusta analyysivaiheesta. Aikajanan tarkoituksena oli selventää ohjausryhmälle mitä toimenpiteitä hankkeessa oli siihen asti tehty ja milloin.



Kuva 28 Aikajana toteutetuista toimenpiteistä analyysivaiheen aikana.

Yksi analyysivaiheen tärkeimmistä päämääristä oli erotella ja määritellä kehitettävät kehitysosa-alueet. Kehitettäviä prosessin osa-alueita pyrittiin havainnollistamaan kuvalla 29. Kuva esittää koko ohjelmiston elinkaarta ja niitä osa-alueita joihin tämä kehityshanke tulee ottamaan kantaa.



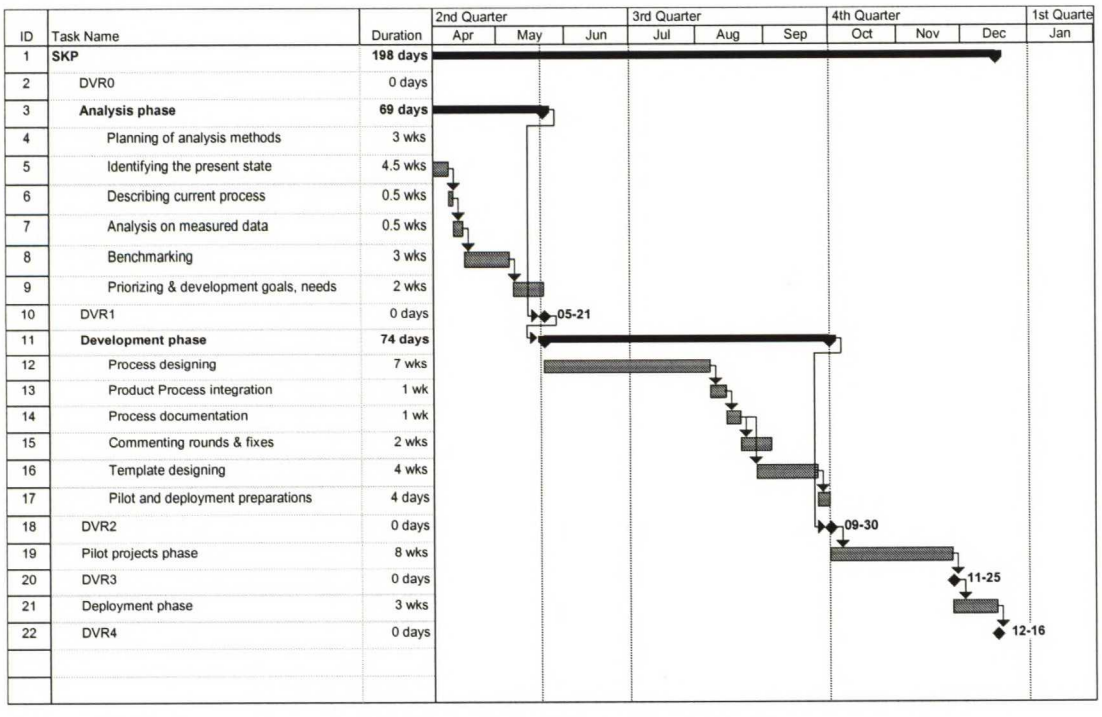
Kuva 29 Ohjelmistoprosessin kehityksen osa-alueet.

Tärkeää oli myös osata rajata pois ne osa-alueet, jotka eivät kuuluneet kehityshankkeen kehityskohteiksi. Rajaamisen teki vaikeaksi se, että käytännössä kuitenkin piti kehittää kaikkia prosessin osa-alueita (vrt. 29). Näitä osa-alueita piti kumminkin osata sisäisesti rajata, jotta kehityshanke pysyisi järkevissä rajoissa. Rajaus tehtiin listaamalla (Taulukko 8) asioita joihin ei tultaisi puuttumaan.

Taulukko 8 Kehityshankkeen ulkopuolelle jätettävät kehitysosa-alueet.

Kehitysalueiden ulkopuolelle on rajattu:
• Turvastandardin IEC 61508 täydellinen noudattaminen
• Käytettävät ohjelmointityökalut
• Ohjelmiston testaustyökalut
• Sovellusohjelmoinnin tukeminen
• Modulaarisen ohjelmistosuunnittelun määrittely

Lopuksi kehityskatselmuksessa käytiin läpi kehitysvaiheen osalta tarkennettua projektin aikataulua (Kuva 30) sekä tarkennettiin kehitysryhmän rooleja. Nyt rooleissa oli selkeästi näkyvissä, että Tuominen on projektin vetäjä, minä kehitys- sekä prosessivastaava ja muut ryhmän jäsenet toimivat ryhmässä oman alansa asiantuntijoina. Ohjausryhmä hyväksyi kehitysryhmän esittämät asiat. Samalla ohjausryhmä antoi luvan suunnittelu- ja kehitysvaiheeseen siirtymiseksi.



Kuva 30 Tarkennettu projektiaikataulu kehitysvaiheen osalta.

7.3 Kehitysvaihe

Vaisalan kehitysprosessi ei määritä tälle vaiheelle kovinkaan tarkkoja ohjeita (Vaisala, 2000). Tuomisen kanssa käydyn esisuunnittelun tuloksena kehitysvaihe päätettiin jakaa noin kuuteen eri työvaiheeseen (kts. Kuva 30). Näistä suurimmat ja työläimmät työtehtävät olivat itse prosessin ja dokumenttipohjien suunnittelu. Suuren työmäärän vuoksi kehitysryhmän yhteisellä päätöksellä viikkopalavereja päätettiin pitää kaksi yhden sijasta.

Kehitysvaiheen aikana korostui ryhmän toiminta. Tässä vaiheessa yritin ottaa enemmän prosessikonsulttimaista otetta ja irtautua itse tehtävään (task) liittyvistä päätöksistä (Schein, 1987). Tähän pyrin antamalla muulle kehitysryhmällä mahdollisimman paljon tietoa sekä tilaa vaikuttaa suunniteltavan prosessiin liittyviin asioihin. Luovuin myös tietoisesti joistakin minun roolini kuuluneista tehtävistä, kuten prosessin puhtaaksi piirtämisestä ja ratkaisupohjien sekä esityksien esisuunnittelusta. Prosessikonsulttimaisen otteen vaikutuksista hankkeeseen pohditaan työn lopussa.

7.3.1 Suunnitteluprosessin kuvaaminen

Ennen ensimmäistä kehitysvaiheen työpalaveria jaoin kaikille kehitysryhmän jäsenille ensimmäisen version tämän diplomityön ohjelmistoprosesseista kertovasta luvusta. Näin pyrin hyödyntämään diplomityötä konkreettisesti kehityshankkeessa ja samalla jakamaan viimeisintä tietoa tiiviissä tietopaketissa.

Kehitysvaiheen alussa oli selvää, että emme pysty hyödyntämään suoraan mitään kirjallisuudesta löytyvää ohjelmiston suunnitteluprosessia. Selkeä linjanveto tehtiin myös sen osalta, että puhtaasti vesiputous- tai agile-tyyliä mukailevaa prosessia ei tulla toteuttamaan. Toteutettavan prosessin nähtiin muistuttavan enemmän hybridimäistä eli vesiputous- ja agile-mallien sekoitusta. Tämän kehitysryhmän yhteisen vision muodostumiseen meni ainakin kehitysvaiheen pari ensimmäistä työpalaveria.

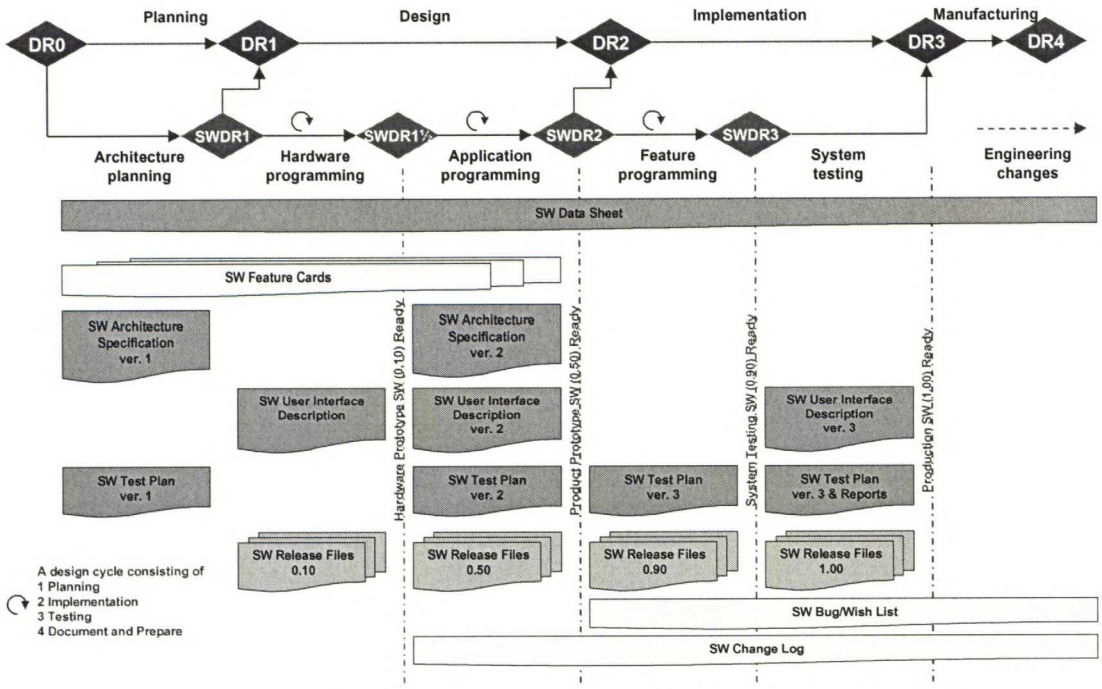
Suunnitteluprosessin kuvaaminen aloitettiin jakamalla ohjelmiston suunnittelu loogisiksi kokonaisuuksiksi sekä selvittämällä keskimääräiset tuoteprojektien katselmuksien väliset pituudet. Analyysivaiheen alussa tai sen aikana kukaan ei osannut jakaa ohjelmiston suunnittelua perustellusti osiin. Nyt kuitenkin jakaminen kehitysryhmässä onnistui ja jopa suhteellisen yksimielisesti, joten analyysivaiheesta oli ollut todellista hyötyä yhteisen näkemyksen aikaansaamiseksi. Yhteisen näkemyksen konkretisoimiseksi piirrettiin siitä ensimmäinen luonnos. Tämä luonnos on esitelty liitteessä (Liite 5, Ensimmäinen kehitysryhmän yhteinen luonnos suunnitteluprosessista).

Lopulta muutamien kokouksien ja useiden keskustelujen jälkeen jokainen vaihe sai oman nimensä ja ne muodostivat kehitysryhmän mielestä loogisen kokonaisuuden. Kehittelyn tuloksena saatu uusi prosessi noudatti sykliajattelua, mutta se lopulta piirrettiin Vaisalassa paremmin tunnettuun muotoon. Sykliajatus jäi prosessin taustalle ja sen nähtiin kuvaavan hyvin jokaisen vaiheen sisällä tehtäviä työvaiheita. Prosessikuvauksen ensimmäisen virallisemman version teki ryhmän vetäjä Tuominen.

Lähes samaan aikaan prosessikuvausta mietittäessä suunniteltiin prosessin aikaisia dokumentteja. Dokumenttien ajoitus suunnitteluprosessiin oli oleellista ja niiden avulla pystyttiin myös tarkemmin määrittämään työvaiheita ja niiden sisältöä. Ennen dokumenttien lopullista määritystä oli kehityshanke ehkä hiukan epätietoisuuden tilassa. Tilanteen laukaisi mielestäni kehitysryhmän jäsen Kokki, jonka henkilökohtaisella ja konkreettisella työpanostuksella dokumentit ja niiden nimet saatiin määriteltyä sekä kirjattua ylös.

Juuri ennen kesälomia saatiin ensimmäinen virallisempi kuvaus prosessista. Tämä prosessi on esitetty kuvassa 31. Tämän prosessin kaikki vaiheet sekä dokumentit kuvattiin sanallisesti prosessimanuaalin muotoon. Tätä työtä helpotti huomattavasti

Tuomisen tekemä ja ideoima pohja, jossa selitykset saatiin kätevästi linkitettyä kuvattuun prosessiin.



Kuva 31 Ensimmäinen virallisempi versio ohjelmiston suunnitteluprosessista.

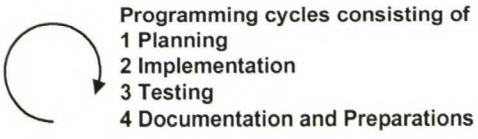
Kehityshankkeen aikataulussa oli huomioitu kesälomat. Lisäksi kehitys oli selvästi aikataulussaan, joten lomat voitiin viettää rauhallisin mielin.

7.3.2 Suunnitteluprosessin sanallinen kuvaus

Sulautetun ohjelmiston elinkaari koostuu kuudesta eri vaiheesta. Elinkaari alkaa DR0 katselmuksesta ja päättyy tuotteen tuotannon lopettamiseen (vrt. Kuva 31). Seuraavassa käydään läpi ohjelmiston elinkaarta kuvaavan prosessin eri vaiheet sekä prosessiin liittyvät dokumentit. Tässä kappaleessa esitetyt sanalliset selitykset löytyvät myös prosessimanuaalista, joka on prosessin julkaisemisen jälkeen kaikkien nähtävissä ja selailtavissa Vaisalan intranetissä.

Sykli

Sykli kuvaa prosessin suunnitteluvaiheiden aikana tehtäviä toimenpiteitä. Sykli koostuu neljästä eri vaiheesta, jotka ovat suunnittelu, toteutus, testaus sekä dokumentointi ja valmistautuminen (Kuva 32).



Kuva 32 Suunnittelusykli ja sen työvaiheet.

Jokainen suunnittelusykli alkaa kyseisen vaiheen työtehtävien suunnittelulla (planning). Tämä pitää sisällään mm. lohkokaavioiden tekemistä, mittausalgoritmien valitsemista, syklin aikana toteutettavien ominaisuuksien selkeyttämistä ja lukuisia muita projektista riippuvaisia esisuunnittelutehtäviä.

Suunnittelutyövaihetta seuraa toteutus (implementation) eli ohjelmointi. Ohjelmoinnin aikana ohjelmoija tai ohjelmoijat tekevät myös omatoimista suunnittelun aikaista testausta.

Kolmantena vaiheena on testaus (testing). Testaus ja sen laajuus riippuvat aina projektin koosta ja vaativuudesta. Syklin aikaisessa testauksessa on pyrittävä testaamaan aina kyseisen syklin aikana toteutetut ominaisuudet. Testisuunnitelmaa sekä -raporttia voidaan käyttää hyväksi myös syklien aikana tehtävien testien dokumentoinnissa.

Jokainen sykli päättyy dokumentointiin (documentation) ja seuraavan prosessin vaiheen valmisteluun (preparations). Dokumentointia tapahtuu myös syklin jokaisessa vaiheessa, mutta tämän työvaiheen tarkoituksena on painottaa, että se on prosessin joka vaiheessa tehtävää jatkuvaa toimintaa. Lisäksi valmistautumisella painotetaan valmistautumista katselmuksiin ja seuraaviin vaiheisiin, jotta osattaisiin ennakoida tulevaa ja prosessi olisi hallittavampi.

Arkkitehtuurisuunnittelu

Arkkitehtuurisuunnittelun (architecture planning) tarkoituksena on muodostaa ja vakiinnuttaa tuoteprojektin projektisuunnitelma sulautetun ohjelmiston osalta sekä tehdä asiakas- ja teknisten vaatimusten avulla ohjelmiston arkkitehtuuridokumentti. Lisäksi aloitetaan ominaisuuskorttien täyttäminen. On huomioitava, että toisin kuin muut tuotevaatimukset, ohjelmiston ominaisuuskortit eivät "jäädy" tämän vaiheen jälkeen.

Tässä ohjelmiston elinkaaren ensimmäisessä vaiheessa perustetaan lähdekoodille ja sen dokumentoinnille prosessimanuaalin mukaisesti tallennuspaikat. Lisäksi luodaan ohjelmiston datalehti, joka tulee toimimaan kyseisin ohjelmiston "kotisivuna" koko sen elinkaaren ajan.

Arkkitehtuurisuunnitteluvaihe päättyy SWDR1-tilaisuuteen. Tässä tilaisuudessa ovat paikalla kaikki osaston ohjelmistosuunnittelijat sekä tarvittaessa testausinsinööri. Tilaisuuden päätarkoituksena on käydä läpi ohjelmiston arkkitehtuurisuunnittelu. Tämän katselmuksen aikana käytävän keskustelun toivotaan tuovan uusia ideoita ohjelmistosuunnittelijoille. Tarkoituksena on lisäksi tarkastella ohjelmiston arkkitehtuurin rakennetta sekä tehdä riskianalyysi.

Rajapintasuunnittelu

Rajapintasuunnitteluvaiheen (hardware programming) päätehtävänä on pyöräyttää yksi sykli ympäri ja saada näin aikaan elektroniikka- ja tarvittaessa anturitestausta tukeva prototyyppi. Tämän vaiheen tuloksena saatava ohjelmaversio on 0.10, josta tehdään myös lyhyt selostus (release note) ohjelmiston datalehteen.

Vaiheen pituus on synkronoitu elektroniikan ensimmäisen prototyyppikierroksen kanssa. Tämän vuoksi voidaan arkkitehtuurisuunnittelua tarkentaa elektroniikan rajapinnan osalta.

Vaiheen päättyminen eli SWDR1½-katselmus järjestetään projektista riippuen noin 1-3 viikkoa elektroniikkaprototyypin saapumisen jälkeen. Tässä tilaisuudessa ovat läsnä ohjelmistosuunnittelija, elektroniikkasuunnittelija, projektipäällikkö ja testausinsinööri. Tilaisuuden tarkoituksena on luovuttaa tehty ohjelmaversio testauskäyttöön sekä samalla toimia ohjelmisto- ja elektroniikkasuunnittelijan yhteistoiminnan parantajana. Ohjelmisto- ja elektroniikkasuunnittelijan tiedonkulun parantamiseksi ohjelmistosuunnittelija tekee ennen palaveria toteutetuista käyttöliittymistä ja komennoista kuvauksen siihen tarkoitettuun dokumenttipohjaan.

Sovellussuunnittelu

Sovellussuunnitteluvaiheen (application programming) tehtävänä on suunnitella ja toteuttaa ensimmäisen laiteprototyypin sulautettu ohjelmisto. Se pyritään toteuttamaan valitsemalla tähän asti tehdyistä ominaisuuskorteista ne kortit, jotka pakottavat suunnittelemaan koko ohjelmiston arkkitehtuurin rungon. Tämän vaiheen aikana on myös viimeinen mahdollisuus lisätä, muuttaa tai poistaa ominaisuuskortteja.

Suunnittelun tueksi on prosessiin kuvattu muutoslogidokumentti. Tämän dokumentin tehtävänä on dokumentoida kaikki suunnitteluvaiheen versioiden välillä tapahtuneet ohjelmistomuutokset. Näin ohjelmoija ja testaajat pysyvät ajan tasalla versioita vaihdettaessa. Myöhemmin dokumentin on ajateltu kuuluvan osaksi versionhallintaohjelmistoa.

Sovellussuunnittelu päättyy SWDR2-tilaisuuteen, jossa ovat läsnä ohjelmistosuunnittelija, projektipäällikkö ja tuotelinjapäällikkö. Tämä katselmus on erittäin tärkeä ohjelmiston kannalta. Tässä tilaisuudessa päätetään ja jäädytetään ne ohjelmiston ominaisuudet, jotka lopputuotteelta halutaan. Lisäksi, samoin kuin edellisenkin vaiheen päätteeksi, tallennetaan uusin ohjelmistoversio uudella numerolla 0.50 ja siitä tehdään datalehteen lyhyt kuvaus (Release note).

Ominaisuussuunnittelu

Kuten nimikin sanoo, ominaisuussuunnitteluvaiheessa (feature programming) suunnitellaan ja toteutetaan viimeiset ominaisuudet, jotka on päätetty SWDR2-tilaisuudessa. Tämän vaiheen eli syklin jälkeen on tuotantoon luovutettavan ohjelmiston oltava valmis systeemitestausta varten. Vaiheen aikana myös viimeistellään testaussuunnitelma sekä luodaan virhe-/toivedokumentti. Virhe-/toivedokumenttiin kirjataan ylös kaikki uudet virheet ja toiveet, jotka ovat tulleet ilmi ohjelmiston ominaisuuksien jäädyttämisen eli SWDR2:n jälkeen. Tämän dokumentin tarkoitus on auttaa ohjelmistosuunnittelijaa ohjelmiston elinkaaren loppuun asti.

Tämä vaihe päättyy ohjelmiston version 0.90 luovuttamiseen systeemitestausta varten. Luovutus tapahtuu SWDR3-tilaisuudessa, johon on kutsuttu ohjelmistosuunnittelijan lisäksi, projektipäällikkö ja testausinsinööri. Tilaisuuden päätarkoituksena on jäädyttää testaussuunnitelma ja keskustella siihen liittyvistä toimenpiteitä.

Systeemitestaus

Systeemitestauksessa (system testing) sulautettu ohjelmisto testataan viimeisimmän laiteprototyypin kanssa sekä korjataan löydetyt virheet. Systeemitestauksen aikana viimeistellään myös käyttöliittymäkuvausdokumentti sekä tehdään testeistä testiraportit. Vaihe päättyy tuoteprosessin DR3-tilaisuuteen, jossa ensimmäinen tuotantovalmis ohjelmaversio 1.00 luovutetaan tuotantoon ja siitä kirjoitetaan lyhyt kuvaus datalehteen.

Ylläpitovaihe

Systeemitestauksen jälkeen siirrytään ohjelmiston ylläpitoon (engineering changes). Ylläpidon alkuvaiheessa voidaan haluttaessa toteuttaa muutoksia, joita on tullut esiin SWDR2-tilaisuuden jälkeen. Nämä muutokset toteutetaan normaalin muutoskäytännön mukaisesti. Muutoksia tehtäessä päivitetään kaikkia niitä prosessin dokumentteja, joihin muutos on vaikuttanut. Ylläpitovaihe kestää ohjelmiston elinkaaren loppuun asti.

Dokumentointi

Edellä kerrotussa prosessikuvauksessa on mainittu kaikki prosessin aikana tehtävät dokumentit. Kaikille dokumenteille on olemassa valmiit pohjat, jolloin dokumentointi on yhdenmukaista. Dokumentit myös tallennetaan prosessimanuaalin ohjeiden mukaisesti hakemistoihin, jolloin ne ovat helpommin löydettävissä.

7.3.3 Dokumenttipohjien suunnittelu

Dokumenttipohjien suunnittelu aloitettiin kesälomien jälkeen. Mielestäni asiantuntijoita ohjaavien prosessien yhteydessä olevat dokumenttipohjat ovat ainakin yhtä tärkeitä kuin itse prosessikuvaus. Pohjien suunnitteluun käytettiinkin tässä hankkeessa suhteellisen paljon aikaa ja resursseja. Niiden suunnittelu osoittautui myös projektin vaativimmaksi ja raskaimmaksi vaiheeksi.

Dokumenttipohjien suunnittelussa korostui ehkä eniten Kosonen et al. kirjassa mainittu 10 kehittäjän käskyn kohta "Älä jää tuleen makaamaan" (Kosonen et al, 1998). Lisäksi huomasin tärkeäksi tekijäksi asioiden konkretisoinnin. Toisin sanoen oli parempi tehdä jokin ehdotus ja esittää sitten se ryhmälle, kuin vain keskustella ryhmässä mitä dokumentissa voisi olla. Kun asian joskus väkisinkin laittoi paperille ja esitteli kehitysryhmälle, muodostui siitä aina kommentteja ja näin päästiin eteenpäin.

Tässä vaiheessa ratkaisevaksi tekijäksi nousi jokaisen kehitysryhmän jäsenen panos kehitystyöhön. Pohjien suunnittelussa pyrittiin myös hyödyntämään ryhmän jäsenten henkilökohtaisia erikoisosaamisia. Näin myös, että tässä vaiheessa minun roolini oli tehdä mahdollisimman paljon konkretisoivaa työtä käyttäen apuna muun kehitysryhmän osaamista.

7.3.4 Organisaation muutos

Tässä vaiheessa kehityshanketta tuli esiin organisaatiomuutos, johon ei oltu varauduttu. Divisioonaamme liitettiin kaksi uutta tuoteperhettä ja niiden mukana tuli uusia

ohjelmistosuunnittelijoita muista Vaisalan divisioonista. Mielenkiintoiseksi tilanteen teki se, että uusien ohjelmoijien keskuudessa oli havaittavissa lievää muutosvastarintaa. Tähän tuntui suurimpana syynä olevan edellisen divisioonan epäonnistunut samantyyppinen ohjelmiston suunnitteluprosessin kehityshanke. Uusien suunnittelijoiden mukaan saamiseen tultiin myös kiinnittämään huomiota. Näistä toimenpiteistä kerrotaan tarkemmin työn loppuosassa.

7.3.5 Kommentointikierrokset

Ohjelmistosuunnittelijoiden ja projektipäälliköiden sitouttamiseksi muutokseen paremmin ehdotin jo kehitysvaiheen työvaiheita suunniteltaessa kehitysryhmälle, kehitysprosessista tosin hiukan poiketen, virallisempien kommentointikierrosten järjestämistä. Näiden tilaisuuksien tarkoituksena olisi antaa osallistujille mahdollisuus sanoa vapaasti mielipiteensä prosessista. Kehitysryhmä hyväksyi ajatuksen ja ne päätettiinkin järjestää tavallisten hankkeen tiedotustilaisuuksien lisäksi. Erityisen tärkeäksi mielestäni kommentointikierrokset teki tapahtunut organisaatiomuutos.

Ensimmäinen kommentointikierros tapahtui jo ennen lomille lähtöä. Päätimme ottaa ensimmäiset kommentit uunituoreesta prosessista ohjausryhmältä sekä kehityspäälliköltä. Tämä tapahtui lähettämällä heille tehty prosessimanuaaliluonnos. Tämän kierroksen tuloksena saimme muutamia hyviä ideoita prosessin parantamiseksi.

Seuraavaksi järjestettiin kommentointitilaisuus ohjausryhmälle. Tässä tilaisuudessa ohjausryhmän kommentit painottuivat lähinnä dokumenttien sisältöön sekä ylläpitovaiheen tehtäviin. Ohjausryhmän kommenttien perusteella tehtiin parannuksia prosessiin. Ohjausryhmä vaikutti tyytyväiseltä saavutettuun tulokseen ja oli yksimielinen hankkeen suunnan oikeellisuudesta.

Kolmas isoin ja ehkä tärkein kommentointitilaisuus järjestettiin auditoriossa. Tilaisuutta markkinoitiin myös nimellä "kommentointitilaisuus". Tähän tilaisuuteen kutsuttiin kaikki ohjelmiston suunnittelijat sekä projektipäälliköt, mutta ei ohjausryhmää. Tilaisuus pyrittiin valmistelemaan hyvin etukäteen: Tilaisuuteen luotiin agenda, määrättiin esiintyjät sekä tehtiin herättelykysymyksiä, jos keskustelua ei muuten syntyisi. Lisäksi Kokin hyvästä ehdotuksesta tilaisuuteen valmisteltiin FAQ eli vastauksia odotetuimpiin kysymyksiin. Näin pyrimme osoittamaan, että prosessia suunniteltaessa olemme miettineet juuri niitä asioita, jotka koskettavat yksittäistä ohjelmistosuunnittelijaa. FAQ-kysymykset vastauksineen ovat liitteenä (Liite 6, FAQ-kysymykset).

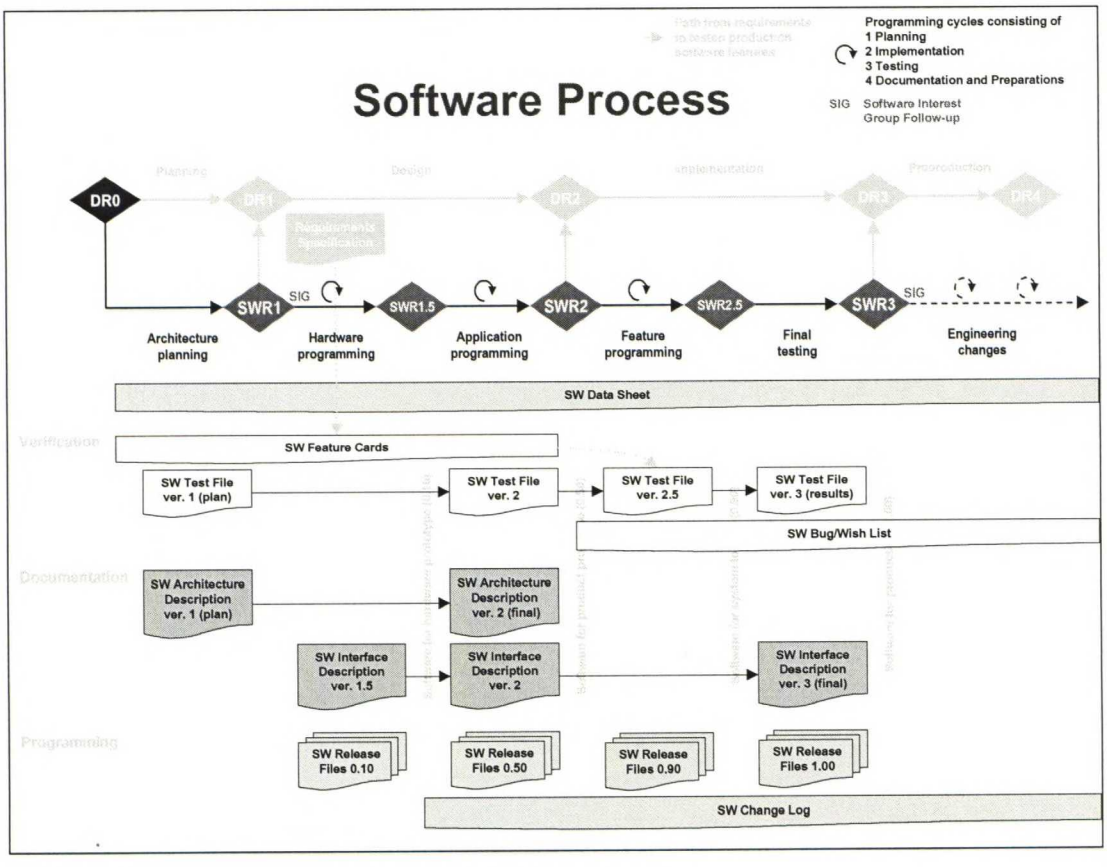
Kommentointitilaisuuden roolien valintaan kiinnitimme erityistä huomiota, jotta esityksestä tulisi mahdollisimman uskottava. Tuominen ja Kokki valittiin esiintyjiksi ja minä toimin kirjurina. Tehtävänäni oli kirjata kaikki tulleet ehdotukset fläppitaulule. Näin pyrittiin osoittamaan osallistujille, että annetut kommentit tulivat huomioiduksi. Lisäksi tällä estettiin keskustelun takertuminen yhteen asiaan, esimerkiksi toteamalla, että asia on kirjattu ylös ja siihen tullaan varmasti ottamaan kantaa.

Tilaisuus meni hyvin ja keskustelu antoi uusia ideoita. Tilaisuuden jälkeen kokosin kommenteista muistion, joka lähetettiin vielä kaikille tilaisuuteen ilmoittautuneille. Muistiossa mainittiin vielä erityisesti, että kaikkiin annettuihin kommentteihin tul-

laan vastaamaan ja että vastaukset tullaan esittämään ennen virallista prosessin koulutusta.

Kommentointien perusteella tehdyt parannukset

Kommentointitilaisuudet sekä sen jälkeen pidetyt kehitysryhmän kokoukset jalostivat kuvattua prosessia lähinnä katselmuksien ja dokumenttien osalta. Uusi jalostunut prosessi on esitelty kuvassa 33.



Kuva 33 Jalostunut ohjelmiston suunniteluprosessi.

Suunnittelijat sekä projektipäälliköt pitivät tärkeänä opitun tiedon jakamista lopputestauksesta sekä projektista. Tämän takia prosessiin lisättiin yksi katselmus DR3-tilaisuuden läheisyyteen, jonka päätarkoituksena on esitellä ohjelmiston intressiryhmälle projekti, syntyneet uudet moduulit sekä testausmetodit. Katselmoinnin lisäys ei kuitenkaan vaikuttanut suunnittelun vaiheistukseen.

Viimeisen vaiheen nimi System Testing muutettiin muotoon Final Testing. Tämä siksi, että System Testing miellettiin usein koko laitteiston lopputestaukseksi. Halusimme kuitenkin korostaa, että ohjelmistoprosessin lopputestauksessa on tarkoituksena nimenomaan testata ohjelmiston kaikki ominaisuudet.

Dokumenttien määrä uudessa sekä vanhassa kuvauksessa on sama, mutta ne ryhmiteltiin paremmin kuvaamaan niiden tarkoituksia. Kommentointikierrokset antoivat myös vahvistuksen sille, että prosessissa on tarvittava määrä dokumentteja yleisimpiin projekteihin.

Dokumenttien kommentointi keskittyi suurimmaksi osaksi niiden sisällön parannusekä muutosehdotuksiin. Lisäksi dokumenttien nimityksiä kommentoitiin ja niitä muutettiinkin paremmin kuvaaviksi. Nimien valintaan vaikutti myös totutut käytännöt ja mieltymykset.

7.3.6 Pilottiprojektien suunnittelu

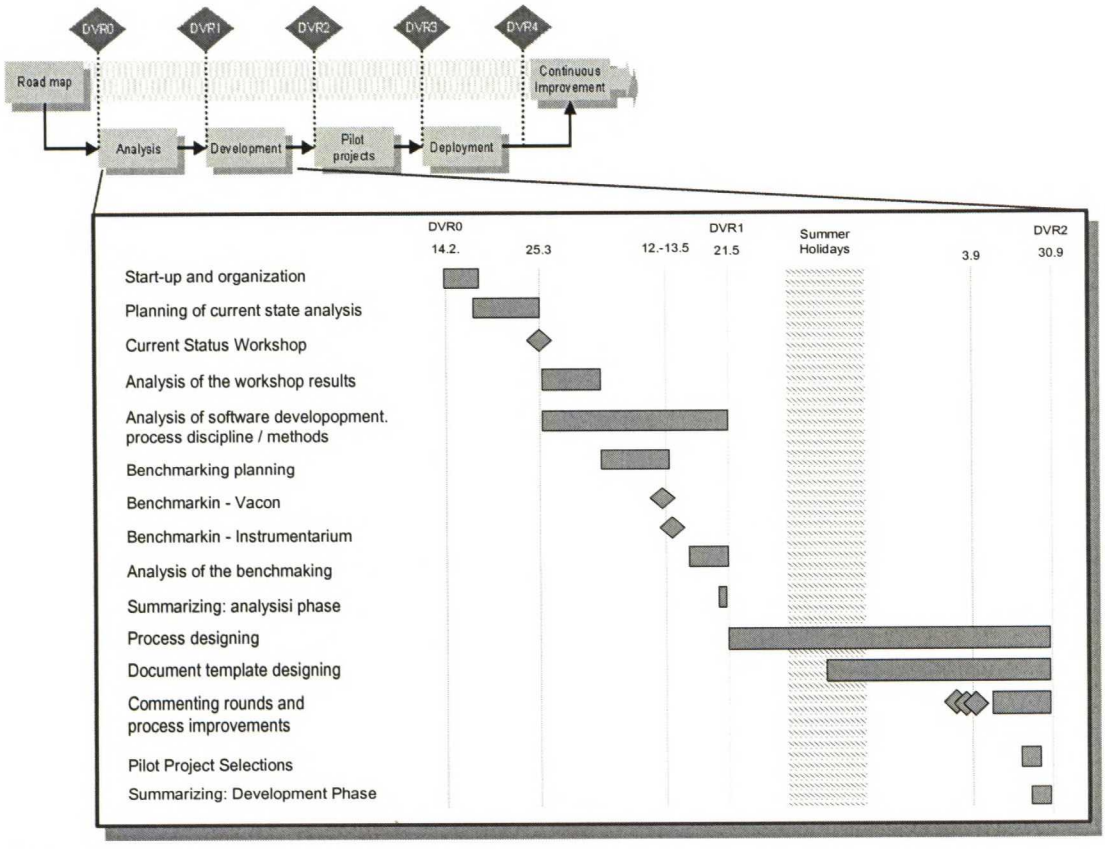
Samalla kun toteutettiin kommentointikierroksilla esiin tulleita parannuskohteita, suunniteltiin jo tulevan pilotointivaiheen toimenpiteitä. Tätä suunnittelua ei olisi voinutkaan tehdä kovin paljoa etukäteen, koska esimerkiksi dokumentit elivät lähes DVR2-tilaisuuteen saakka. Perusperiaate pilottivaihetta suunniteltaessa oli saada testattua kaikki dokumenttipohjat sekä osa katselmuksista. Tämä toteutettiin selvittämällä kaikki meneillä olevat laite-/ohjelmistoprojektit ja niiden sen hetkinen valmiusaste. Näiden tietojen perusteella muodostettiin suunnitelma. Taulukossa 9 on esimerkin vuoksi esitelty dokumenttipohjien pilotointisuunnitelma, samanlainen suunnitelma tehtiin myös kehityskatselmuksien osalta.

Taulukko 9 Pilotointisuunnitelma (1.10 - 26.11.2003).

Project Status	Project Phase	Piloted SW-Process Phase	Piloted Document Template	The Pilot	Status
All	All	All	Data Sheet	All	?
All	All	All	Release Files	All	?
PROJ01	DR0-1	Architecture Plannig	Arch. Desc.; Feat. Cards;Test Files	N.N.	?
PROJ02	DR0-1	Architecture Plannig	Arch. Desc.; Feat. Cards;Test Files	N.N.	?
PROJ03	DR1-2	Application Prog.	Arch. Desc.; User Interface	N.N.,N.N.	?
PROJ04	DR1-2	Application Prog.	User Interface	N.N.	?
PROJ05	DR2-3	Feature Prog.	Test Files	N.N.	?
PROJ06	DR3-4	Engineering Changes	Test Files; Change Log.	N.N.	?
PROJ07	DR3-4	Engineering Changes	Change Log.; Bug/Wish List	N.N.	?
PROJ08	DR3-4	Engineering Changes	Change Log.; Bug/Wish List	N.N.	?

7.3.7 Kehityskatselmus 2 (DVR2)

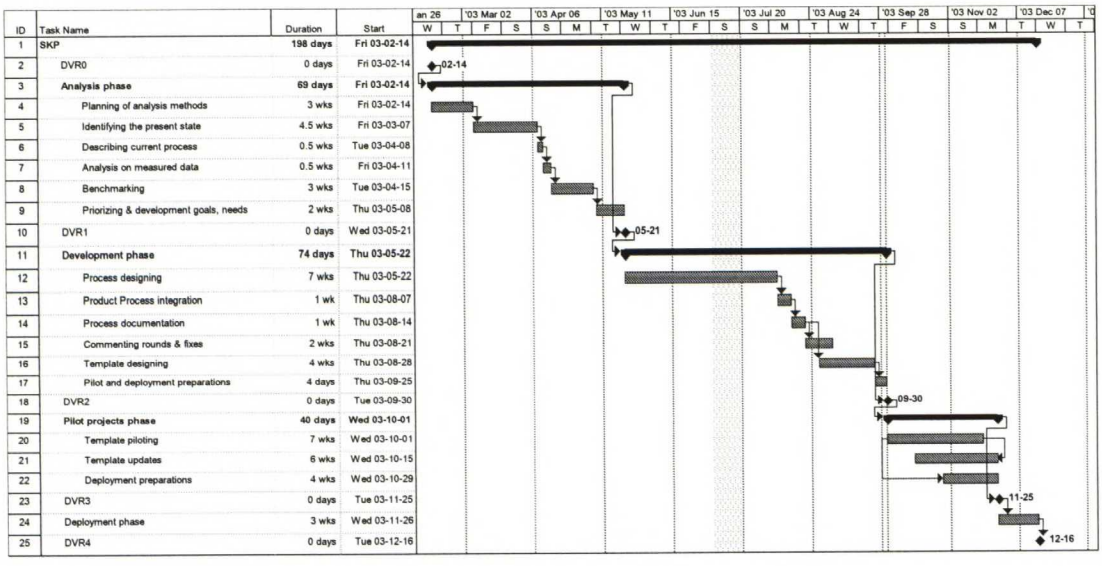
Kehitysvaihe päättyi DVR2-tilaisuuteen. Sen tarkoituksena oli käydä läpi kehitysvaiheessa saadut tulokset. Kuva 34 havainnollistaa niitä toimenpiteitä, joita hankkeessa oli tehty ennen DVR2-tilaisuutta.



Kuva 34 Aikajana toteutetuista toimenpiteistä kehityshankkeen aikana.

DVR2-tilaisuus aloitettiin esittelemällä viimeisin versio prosessista. Koska prosessin kulku oli jo kommentointikerroksien ansiosta kaikille tuttu, keskityttiin ohjausryhmän pyynnöstä dokumenttipohjien tarkasteluun. Tarkastelun tuloksena löydettiin muutamia epäkohtia sekä parannusehdotuksia. Nämä kirjattiin ylös ja päätettiin korjata ennen pilottiprojektien alkua.

Tilaisuudessa esitettiin myös pilottiprojektisuunnitelmat (kts. taulukko 9), jalkauttamisvaiheen alustava suunnitelma sekä vastuukysymykset. Ohjausryhmältä saatiin myös lisäohjeeksi, että olisi hyvä pilotoida kaikki katselmukset vaikka niihin ei olisikaan täytettyjä dokumentteja. Tilaisuuden lopuksi esiteltiin kuvan 35 päivitetty projektisuunnitelma.



ID	Task Name	Duration	Start
1	SKP	198 days	Fri 03-02-14
2	DVR0	0 days	Fri 03-02-14
3	Analysis phase	69 days	Fri 03-02-14
4	Planning of analysis methods	3 wks	Fri 03-02-14
5	Identifying the present state	4.5 wks	Fri 03-03-07
6	Describing current process	0.5 wks	Tue 03-04-08
7	Analysis on measured data	0.5 wks	Fri 03-04-11
8	Benchmarking	3 wks	Tue 03-04-15
9	Priorizing & development goals, needs	2 wks	Thu 03-05-08
10	DVR1	0 days	Wed 03-05-21
11	Development phase	74 days	Thu 03-05-22
12	Process designing	7 wks	Thu 03-05-22
13	Product Process integration	1 wk	Thu 03-08-07
14	Process documentation	1 wk	Thu 03-08-14
15	Commenting rounds & fixes	2 wks	Thu 03-08-21
16	Template designing	4 wks	Thu 03-08-28
17	Pilot and deployment preparations	4 days	Thu 03-09-25
18	DVR2	0 days	Tue 03-09-30
19	Pilot projects phase	40 days	Wed 03-10-01
20	Template piloting	7 wks	Wed 03-10-01
21	Template updates	6 wks	Wed 03-10-15
22	Deployment preparations	4 wks	Wed 03-10-29
23	DVR3	0 days	Tue 03-11-25
24	Deployment phase	3 wks	Wed 03-11-26
25	DVR4	0 days	Tue 03-12-16

Kuva 35 Tarkennettu projektiaikataulu pilotointivaiheen osalta.

7.4 Pilotointivaihe (Pilot Projects)

Pilotointivaihe käynnistyi kehitysryhmän palaverilla, jossa lopullisesti lyötiin lukkoon pilotointimenetelmät sekä ryhmän jäsenten vastuualueet. Tämän jälkeen aloitettiin noin kaksi kuukautta kestävä pilotointivaihe.

Dokumenttipohjat pilotoitiin siten, että jokainen ryhmän jäsen toimi henkilökohtaisena tukihenkilönä pilotointia tekevälle suunnittelijalle. Näin pystyttiin keräämään välitöntä palautetta esimerkiksi dokumenttipohjien ongelmakohdista sekä muista niihin liittyvistä käytännön asioista.

Kaikki kehityskatselmukset katselmoitiin, vaikka joskus jotkut vaaditut dokumentit puuttuivatkin. Tämä siksi, että olisi ollut kohtuutonta vaatia melkein valmiin projektin suunnittelijaa kirjoittamaan dokumentteja vain pilotointia varten. Lisäksi prosessin omistajan Rantasen kokemuksen mukaan, on pilotoinnissa yleensä ollut hyödyllistä edes istua suunnitellussa kokoonpanossa ja ihmetellä yhdessä onko dokumenttipohjissa ja kokouksessa ideaa.

7.4.1 Pilotointivaiheen tulokset

Pilotointivaiheen tulokseksi saatiin paljon parannusehdotuksia dokumenttipohjien sisältöön. Lisäksi katselmuksien pilotoinnin ansiosta niiden sisältöä sekä osallistujia pystyttiin tarkentamaan. Kerättyjä kommentteja ja huomioita analysoitiin kehitysryhmän viikkopalavereissa. Näissä palavereissa sovittiin myös prosessiin ja dokumenttipohjiin tehtävistä muutoksista sekä seurattiin pilotoinnin edistymistä.

Pilottivaiheen loppuvaiheessa alettiin jo valmistautua jalkauttamisvaiheeseen sekä DVR3-tilaisuuteen. Tätä varten suunniteltiin mm. koulutustilaisuus sekä siirrettiin prosessin ohjeistuksia ja prosessimanuaali intranetiin.

7.4.2 Kehityskatselmus 3 (DVR3)

Pilotointivaihe päättyy DVR3-katselmukseen. Tämän tilaisuuden tarkoituksena on mm. katselmoida pilottivaiheen tulokset ja esitellä toimintasuunnitelma jalkauttamisvaiheelle. DVR3-tilaisuus pidetään marraskuun lopulla eli tämän diplomityön palauttamisen jälkeen.

8 YHTEENVETO JA JOHTOPÄÄTÖKSET

Tässä luvussa vedetään yhteen saadut tulokset ja verrataan niitä työn alussa esitettyihin teorioihin sekä toimenpide-ehdotuksiin. Luvun tarkoitus on myös vastata tutkimusongelmaan etsimällä kriittisiä tekijöitä, jotka vaikuttivat työn lopputulokseen.

8.1 Kehityshankkeen kulku ja onnistuminen

Voidaan sanoa, että kehityshankeen kulku noudatti kuvan 2 kehitysprosessia. Tässä kappaleessa kerrotaan mitkä olivat kehityshankkeen kulun kannalta kriittisimmät ja merkittävimmät tapahtumat ja miten kehittämisen teoriaa pystyttiin hyödyntämään työssä.

Vision kommunikointi

Kehityshankkeella oli alusta lähtien Instrumentsin tuotekehityksen johdon tuki, joka esimerkiksi Kotterin mukaan on yksi muutoksen menestymisen kriittisistä tekijöistä. Johdon näkemystä tukemaan tehdyt kompetenssi sekä tiekarttasuunnittelut auttoivat johtoa myös vision kommunikoinnissa muulle organisaatiolle. Johto lisäksi sitoutui itse sekä se osasi sitouttaa henkilöstöä muutokseen osallistamalla heitä strategiasuunnitteluihin. Johdolle oli myös alusta lähtien selvää, että se haluaa parantaa vain ohjelmiston suunnitteluprosessia, joka myöhemmin helpotti kehityshankeen aikana tehtyjen kehityskohteiden rajaamisia. (Denton, 1996; Kotter, 1995; Lanning et al, 1999.)

Tärkein tekijä hankkeen vision läpiviemisessä oli varmasti tiekartta. Tuomisen hyvin kokoama ja hahmotelma tiekartta esiteltiin strategiapalaverissa, jossa se toi kaikille ohjelmiston suunnittelijoille hyvän kokonaiskuvan tästä kehityshankkeesta ja siitä miten se liittyy suurempaan kokonaisuuteen. Näin sen avulla pystyttiin kommunikoimaan ja konkretisoimaan tavoitteita, jotka ovat tärkeitä tekijöitä muutoksien onnistumiseksi (Kotter, 1995; Lanning et al, 1999).

Kehitysryhmä ja roolini siinä

Hankkeen tärkein työrukkanen oli kehitysryhmä. Tätä ryhmää valvoi sekä ohjasi ohjausryhmä. Ryhmien välinen yhteistyö sujui koko hankkeen aikana hyvin. Ohjausryhmä antoi kehitysryhmälle tarvittavan määrän vapauksia sekä päätäntävaltaa valtuuttamalla (Kotter, 1995). Ohjausryhmän ei tarvinnut hankkeen aikana puuttua oleellisesti kehitysryhmän toimintaan, joten sen pääasialliseksi tehtäväksi jäi väliaikaraporttien seuraaminen.

Kehityshankkeen alku ts. analyysivaiheen aloittaminen ei ollut helppoa. Kehitysryhmä ei ollut ennen työskennellyt yhdessä eikä kukaan ryhmästä ollut tehnyt vastaavanlaista hanketta. Hankkeen alussa suurimpana ongelmana oli otteen saaminen kehitettävästä asiasta. Kaikille oli kyllä selvää mikä visio oli, mutta selkeää toimintatapaa tai strategiaa siihen pääsemiseksi ei tiedetty. Kehityssuunnitelmaa luotaessa toimi apuna kuvan 2 kehitysprosessi. Prosessi selkeytti vaiheet, mutta se ei yksiselitteisesti kuitenkaan kertonut mitä vaiheissa kulloinkin tulisi tehdä.

Tätä yleistä epäselvyyttä pienentämään valitsin aluksi enemmän ryhmän toimintaa ohjaavan prosessikonsulttimaisen otteen. Tavoitteenani oli auttaa ryhmää ongelmassa eteenpäin tekemällä diagnostisia kysymyksiä (diagnostic interventions). Tällä pyrin saamaan heidän näkemyksensä esiin ja samalla kuuntelin heidän ideoitaan tilanteen ratkaisemiseksi. Pyrin myös tällä toiminnalla pitämään vastuun heidän harteillaan ja välttämään ns. ekspertin roolia. Tällä tavoin kehitysryhmä myös sitoutui paremmin tehtyihin päätöksiin. Kun mielestäni sen hetkinen tilanne oltiin saatu kartoitettua sekä hankkeesta saatu jonkinmoinen ote, siirryin enemmän vaihtoehtoja antavaan (action alternative interventions) sekä uskoa luoviin (confrontive interventions) kysymyksiin ja ehdotuksiin. Tällaisten kysymysten ja ehdotusten avuilla tehtiin yhteisiä päätöksiä esimerkiksi kehityssuunnitelmista, analyysimetodeista yms. toimenpiteistä. Totuuden nimissä täytyy kuitenkin todeta, että täysin puhdasoppista Scheinin tekniikkaa en pystynyt noudattamaan ja siihen tuskin kukaan pystyy. Hankkeen edetessä lähinnä huomasin, että henkilökohtaiset kyvyt sekä improvisointitaidot olivat teoriaoppeja tehokkaampia menetelmiä. (Schein, 1987.)

Nykytilan analysointi

Tässä hankkeessa panostettiin nykytilan analysointiin. Tämä siksi, että ohjelmiston suunnittelun nykytilaa ei oltu tätä ennen mitenkään kuvattu ja sen toiminta oli erittäin kulttuurisidonnaista. Kehityshankkeen alkutilanne oli mielestäni erittäin haastava ja hanketta oli mukava lähteä viemään eteenpäin.

Nykytilan analysointia suunniteltaessa pidin tärkeänä lopullisen asiakkaan eli ohjelmiston suunnittelijoiden osallistamista sekä heidän omien näkemysten esilletuomista. Tämä siksi, että analysointivaiheen tarkoituksena on viestiä organisaatiolle muutostarpeesta, sitouttaa heitä muutokseen, kyseenalaistaan nykykäytäntöä ja saada esiin todelliset muutostarpeet (Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999; Vaisala, 2000). Mielestäni järjestetty workshop palveli tätä tarkoitusta erittäin hyvin.

Workshop-tilaisuuteen osallistuivat kaikki suunnittelijat. Siellä he pystyivät kertomaan omia mielipiteitään ensiksi vapaasti kyselylomakkeen avulla, sitten jalostamaan ideaa parin kanssa ja lopuksi hahmottelemaan ryhmissä nykyisen käytännön. Kun saatuja tuloksia vielä tilaisuuden lopuksi analysoitiin saatiin, aikaan tilanne, jossa nykykäytäntöä kyseenalaistettiin ja tarve muutokseen sekä yhteiseen käytäntöön tuli myös suunnittelijoiden taholta. Tämän toivon auttavan prosessin jalkauttamista sekä vähentävän muutosvastarintaa.

Benchmarking

Workshopin ja nykytilan analysoinnin jälkeen seuraava merkittävä tapahtuma oli benchmarking. Benchmarkingilta odotettiin todella paljon, joten se myös pyrittiin hoitamaan tunnollisesti. Lopputulos benchmarkingista olikin kaikkien mielestä antoisa ja hyvä. Lisäksi kehityshankkeen kannalta tämä oli ehkä ratkaisevin käänne. Benchmarking loi kehitysryhmään uskoa sekä toi valtavan määrän uusia ideoita. Esimerkiksi ennen benchmarkingia kehitysryhmän enemmistön mielipide oli, että ainut oikea tapa hoitaa ohjelmiston suunnittelu on perinteinen vesiputousmalli. Benchmarkingin jälkeen kukaan ei enää yhden varoittavan esimerkin jälkeen halunnut tehdä kankeaa vesiputousta vaan muutoshalu ketterämpään tapaan oli silmiinpistävää.

Benchmarkingin jälkeen pysähdyin myös hetkeksi miettimään prosessien kuormituksien välisiä eroja (Kuvat 25 ja 27) ja niiden vaikutuksia työn tulokseen. Kuten Teikarikin on todennut, on tuottavuus työolosuhteiden, työn sisällön, työtaidon ja johtamisen tulo (Teikari, 2002). Toisin sanoen suunnittelemalla huono ja kuormittava prosessi aiheutetaan turhaa kuormitusta ja sitä kautta työtehon laskua. Määrällinen ylikuormittaminen voi myös pitemmällä tähtäimellä altistaa työuupumukselle ja sen laukaisimena voi toimia esimerkiksi töiden kasautuminen projektin loppua kohden, (vrt. kuva 25) (Kärnä & Aro, 2002; Cooper, 1998; Sutherland & Cooper, 2000). Ohjelmistotuotannossa ja erityisesti IT-alalla on Suomessa havaittu viimeaikoina suurtakin työuupumusta (Mäkinen, 2002). Saman ongelman muuallakin maailmassa on havainnut myös esimerkiksi XP-metodin kehittäjä Beck. Tämän estämiseksi hän XP-metodissaan painottaa, että kahta peräkkäistä yli 40h työviikkoa ei saa tulla tai muuten prosessissa, suunnittelussa tai aikataulutuksessa on vikaa. (Beck, 2000.) Näiden seikkojen takia prosessia suunniteltaessa kuormittamiseen kiinnitettiin myös huomiota. Uskon, että kuormittavuudella on myös selkeitä vaikutuksia prosessin jalkauttamisen onnistumiseen.

Välitavoitteet

Pitkissä hankkeissa ovat välitavoitteet erittäin tärkeitä (Lanning et al, 1999). Mielestäni erittäin hyvä välitavoite oli DVR1. Tämä tilaisuus pakotti vetämään yhteen koko analyysivaiheen aikana tehdyt toimenpiteet ja analysoimaan niitä kokonaisuutena. Lisäksi DVR1 pakotti kehitysryhmän rajaamaan kehitettävät osa-alueet sekä tarkentamaan suunnitteluvaiheen aikataulua. Samalla myös ryhmän visio kirkastui olennaisesti.

Kaikki välitavoitteet loivat myös tiettyä uskoa kehitysryhmään, koska ohjausryhmä muodollisesti hyväksyi tehdyt toimenpiteet ja jatkosuunnitelmat. Ne olivat myös tärkeää siksi, että niissä ohjausryhmä olisi voinut tarvittaessa korjata kehitysryhmän toimintaa oikeaan suuntaan. (Lanning et al, 1999; Vaisala, 2000).

Prosessin kehittäminen ja suunnittelu

DVR1-tilaisuuden jälkeen tein tietoisen muutoksen prosessikonsulttimaisessa otteessani. Pyrin irrottautumaan vielä enemmän tehtävään liittyvistä päätöksistä. Tämä oli tietysti vaikeaa, koska omasin eniten teoriatietoa ohjelmistoprosesseista ja niiden kehittämisestä. Toisaalta tämä myös salli tietyn tyyppisen tilan antamisen. Tämän toteutin esimerkiksi siten, että pyrin entistä enemmän toimimaan tietolähteenä enkä päätöksen tekijänä. Esimerkiksi toimitin tämän työn ohjelmistoprosesseja käsittelevän luvun kehitysryhmälle luettavaksi ennen ensimmäisiä prosessin suunnittelupalavereja. Näin toivoin heidän itse muodostavan kuvan parhaasta mahdollisesta tavasta suunnitella ohjelmistoa, koska ko. luvun olen pyrkinyt kirjoittamaan mahdollisimman puolueettomasti ja laajoja lähteitä käyttäen. Päätöksentekotilanteita pyrin puolestaan ohjaamaan tarvittaessa oikeaan suuntaa tarjoamalla kehitysryhmälle eri näkemyksiä asioista.

Suunnitteluvaiheessa tarkoituksenani oli myös aina tarvittaessa luopua ennen rooliini kuuluvista prosessin suunnittelutehtävistä. Tällä tavoin pyrin sitouttamaan kehitysryhmää paremmin suunniteltuihin asioihin ja sitä kautta muutokseen. Näin myös tässä vaiheessa, että hankkeen tulevaisuuden sekä jalkauttamisen kannalta olisi varmasti

tärkeämpää kehitysryhmän muiden jäsenten sitoutuminen tehtyihin ratkaisuihin. Luopuminen tietyistä tehtävistä antoi myös minulle enemmän aikaa keskittyä organisaation kehittämiseen, kokonaisuuden hahmottamiseen sekä itse suunnitteluprosessiin.

Esimerkkinä edellisestä voisi mainita kehitysryhmään kuuluneen ohjelmistosuunnittelija Kokin henkilökohtaisen työpanoksen. Tämä tapahtui kun ensimmäisten prosessiluonnosten jälkeen kehityshanke eli ns. pysähtyneisyyden aikaa. Puhtaaksi piirtämäni prosessikuvaukset alkoivat tyydyttää lähes kaikkia ryhmän jäseniä, mutta silti tuntui että jotain puuttui. Tämän tilanteen ratkaisi mielestäni juuri Kokin henkilökohtainen työpanostus, kun hän teki ehdotelman tarvittavista dokumenteista, niiden sisällöistä ja ajoittumisesta prosessiin. Ryhmä jalosti tätä ehdotusta yhdessä ja näin päästiin iso hypähdys hankkeessa eteenpäin ja saavutettiin kaikkia tyydyttävä lopputulos, Tärkeää tässä oli mielestäni myös se, että tekijänä oli ohjelmiston suunnittelija, joka tulevaisuudessa tulee itsekin sitä käyttämään.

Ryhmä vetäjä Tuominen myös omalla panoksellaan vei hanketta samoihin aikoihin eteenpäin. Hän piirsi oman näkemyksensä mukaan ensimmäisen virallisemman version prosessiluonnoksista. Lisäksi hän teki piirretystä prosessista sellaisen, jota muiden oli helppo täydentää. Tässä vaiheessa päätin luopua prosessin puhtaaksi piirtämisestä, joka oli tätä ennen kuulunut automaattisesti minun vastuulleni. Näin sain itselleni aikaa, ja mikä tärkeintä prosessin kuvaus ei ollut enää yksin minun harteillani. Laamanen on myös kirjassaan todennut, että johtajien pitäisi itse piirtää prosessikuvaukset ja mielestäni tärkein syy tähän on juuri se, että näin he itse vastaavat koko luonnoksesta (Laamanen, 2001).

Ratkaisevaksi kehitysvaiheen kriittiseksi tekijäksi voisi määrittää Kosonen et al. kirjassa olleen seitsemännen kehittäjän käskyn: "Älä jää tuleen makaamaan" (Kosonen et al, 1998). Sillä tarkoitetaan asioiden konkretisointia sekä joskus suoraviivaista toimimista tilanteissa, jossa hanke tuntuu pysähtyneen. Tämä oli mielestäni ainakin tämän hankkeen kohdalla jälkeenpäin ajateltuna ratkaiseva tekijä.

Hyvin ja huolella tehty analyysivaihe palkitsi suunnitteluvaiheessa. Nimittäin kuvatut nykykäytännöt ja sitä kautta saatu yhteinen käsitys selkeytti kehitysryhmän näkemystä asiasta. Tästä ehkä paras esimerkki oli se, että suunnittelu osattiin loppujen lopuksi jakaa loogisiksi kokonaisuuksiksi, vaikka hankkeen alussa se tuntui lähes mahdottomalta.

Kommentointikierrokset

Kommentointitilaisuudet olivat erittäin antoisia. Uskon myös, että kaikilla tilaisuuksilla oli suuri vaikutus kehitysryhmän itseluottamukseen sekä osallistujien sitouttamiseen. Erityisen antoisa oli ohjelmistokehittäjille sekä projektipäälliköille järjestetty tilaisuus. Kommentoinnin tehokkuutta vielä mielestäni lisäsi niiden näkyvä ylöskirjaaminen, sekä niihin myöhemmin vastauksien antaminen. Tilaisuuksien onnistumiseen puolestaan varmasti vaikutti niiden hyvä etukäteisvalmistelu sekä tilaisuuden esiintyjien roolien valinta.

Eräässä kommentointitilaisuudessa tein mielestäni myös tärkeän havainnon. Kun ohjelmistoprosessia arvosteltiin, niin sitä puolustamaan ei enää tarvittukaan kehitys-

ryhmän jäseniä, vaan sen tarpeellisuutta sekä oikeellisuutta puolustivatkin muut osallistujat. Toivotaan, että tämä olisi enne osallistamisen vaikutuksista.

Organisaatiomuutos

Kommentointitilaisuudessa nousi esiin myös ennen erillään olleiden organisaatioiden käytäntöjen yhteensovitusongelma. Koska eroavaisuuksia jo peruskysymyksistä oli erittäin paljon, päätettiin pitää asian puitteissa erillisiä kokouksia. Näiden kokousten avulla pystyttiinkin löytämään keinoja yhteensovittamisen ratkaisemiseksi. Tärkein lopputulos tämän kehityshankkeen kannalta oli yhteisymmärrys ohjausryhmän kanssa siitä, että organisaatioiden toimintatapojen yhdistäminen ei ole tämän kehityshankkeen vastuulla. Toisin sanoen kehityshankkeelle annettiin lupa jatkaa niin kuin oli alunperinkin suunniteltu.

Pilotointi

Pilotoinnin tärkeyden merkitystä ei voi vähätellä. Sen aikana nähtiin ensimmäisen kerran, kuinka suunnitellut asiat todella toimivat käytännössä. Pilotointi myös varmasti lisäsi ohjelmoijien sitoutumista, koska heillä oli näin mahdollisuus suoraan vaikuttaa esimerkiksi dokumenttipohjien sisältöön. Se miten pilotointivaihe todella onnistui ja mitkä olivat sen vaikutukset kehityshankkeen onnistumiseen, tullaan näkemään vasta jalkauttamisvaiheen aikana. Kuitenkin voidaan jo nyt sanoa, että pilotointi sekä kommentointikierrokset toivat heti konkreettisia hyötyä parannusehdotuksien muodossa. Tämä on myös osoitus siitä, että vaikka kuinka kehitysryhmä pyrkii miettimään kaikkia näkökulmia, niin aina jää jotain huomaamatta. Nämä pienet yksityiskohdat voivat myös loppujen lopuksi olla niitä kriittisiä tekijöitä, jotka mahdollistavat tai pilaavat hankeen onnistumisen.

Jalkauttamiseen vaikuttavia tekijöitä

Kommunikointi ja tiedottaminen ovat tärkeässä roolissa muutoksen aikaansaamiseksi (Kosonen et al, 1998; Kotter, 1995; Lanning et al, 1999). Tämä pyrittiin pitämään mielessä koko kehityksen ajan. Esimerkiksi joka kuukausi järjestettävissä ohjelmistosuunnittelijoiden yhteisissä tilaisuuksissa kerrottiin kehityshankkeen sen hetkinen tilanne ja mitä seuraavaksi tullaan tekemään. Lisäksi ohjausryhmää pidettiin ajan tasalla käytäväkeskustelujen sekä sovittujen tapaamisten avulla. Tiedottamisen ja kommunikoinnin tulokset tai niiden puutteet tulevat viimeistään vastaan jalkauttamisvaiheessa. Tämän takia niiden lopullista onnistumista ei voida tämän työn puitteissa enempää arvioida.

Hankkeen aikana yksi mielenkiintoisimmista asioista oli seurata käsitteiden ja erityisesti niiden merkityksien ja käytön muuttumista. Muutoksen ja kehityksen voidaan siis katsoa syntyneen sosiaalisen konstruktion kautta. Esimerkiksi sanat ohjelmistoprosessi ja sykli saivat kehityshankkeen aikana uudet merkitykset ja muodostui tapa jolla niitä voidaan hyödyntää Vaisala Instrumentsin tapauksessa. Tämän näkee konkreettisesti esimerkiksi siitä, että sykli piirrettiin prosessikuvaukseen kuvaamaan työskentelyä prosessin eri vaiheissa, vaikka sanaa sykli tuskin kukaan todella tunsi ennen hankkeen alkua.

Kriittiset tekijät ja tulevaisuus

Tässä kappaleessa on tiiviisti esitetty tämän projektin tärkeimpiä vaiheita ja tekijöitä. Koska projekti jatkuu vielä on vaikea sanoa, mitkä näistä ovat kaikista tärkeimpiä. Tulevaisuuden toimenpiteet ja se, kuinka hanketta sekä kriittisiä tekijöitä tullaan myöhemmin arviomaan, esitellään tarkemmin luvussa 9 .

8.2 Ohjelmiston suunnitteluprosessin sisältö

Tämän kappaleen tarkoituksena on verrata suunniteltua prosessia ohjelmistotuotanto- sekä yleiseen prosessikirjallisuuteen. Tarkoituksena on löytää niitä tekijöitä, jotka tekivät suunnitellusta prosessista juuri Vaisala Instruments -divisioonalle sopivan prosessin.

Kosonen et al. mainitsevat, että kehityksen tuloksien vertaaminen ideaalimalleihin ei ole kannattavaa, koska yleensä aina saavutettu tulos on ideaalimallia huomattavasti laihempi (Kosonen et al, 1998). Tämän takia ideaalitilaan vertailu tapahtuukin vain tämän työn puitteissa ja muuten kehitykseen liittyvissä tilaisuuksissa asioita verrataan alkutilaan.

Suunnitellun prosessin yhtäläisyydet teorian kanssa

Suunniteltu prosessi ei ole suora johdannainen mistään ohjelmistotuotantoalan kirjallisuudessa esitetyistä malleista tai metodeista. Prosessi on kuitenkin lähempänä suunnitelmaohjautuvia kuin agile-tyyppisiä metodeja. Syitä siihen miksi prosessista ei tehty kirjallisuutta noudattavaa ja nykytrendin mukaisesti agile-tyyppistä on useita. Yksi suurimmista syistä oli ehdottomasti se, että suunnitellun prosessin täytyi toimia tuoteprosessin aliprosessina, joka puolestaan on erittäin vesiputoustyyppinen. Tästä syntyvä lähes mahdoton vesiputous-agile yhteensovitusongelma tiedostettiin jo hankkeen alkuvaiheessa, koska esimerkiksi saman tyyppisiä yhteensovitusongelmia havaitsi Grenning yrittäessään implementoida XP-metodia vesiputousmaiseen prosessiympäristöön (Grenning, 2001). Toinen suuri syy oli se, että Vaisala Instruments valmistaa ohjelmistoa, elektroniikkaa, antureita sekä mekaniikkaa sisältäviä laitteita eikä tee esimerkiksi pelkästään sovellusohjelmistoa, joihin agile-metodit ovat lähinnä suunnattu.

Jos kerran suunniteltu prosessi on yhdistelmä kaikista metodista, niin mitä yhtäläisyyksiä voidaan löytää? Silmiinpistävin yhtäläisyys on vesiputous, joka muodostuu SWDR-katselmuksista (kts. Kuva 31). Toisaalta sykliajattelu jäi erittäin voimakkaasti elämään, joten sitä päätettiin käyttää kuvaamaan eri vaiheiden sisältämiä työvaiheita. Syklit olivatkin uutta koko Vaisalan organisaatiossa ja ne tuovat mielestäni yksinkertaisella tavalla selvästi enemmän informaatiota prosessikuvaukseen ja sitä kautta parannusta prosessin omaksumiseen ja noudattamiseen. Lisäksi sykli mielestäni selkeyttää työvaiheita yksin ohjelmoijalle sekä tarvittaessa se tukee pari- tai ryhmäohjelmointia.

Suunnitteluvaiheille ja sykleille pyrittiin saamaan omat tärkeät tehtävät. Tämä siksi, että esimerkiksi ASD-metodissa painotetaan sykleille asetettavien järkevien tavoitteiden asettamista, koska ne ohjaavat syklin aikana tehtäviä ratkaisuja oikeaan suuntaa. Hyvänä suunnan näyttäjänä voidaan Highsmithin mukaan käyttää esimerkiksi

syklien järkevää nimeämistä. (Highsmith, 1999). Analyysivaiheesta saadun kokonaiskuvan ansiosta prosessi saatiinkin jaettua loogisiksi osasiksi, joille pystyttiin asettamaan erittäin hyvät tavoitteet ja kuvaavat nimet (kts. Kuva 31 ja kappale 7.3.2).

Teoriakatsauksen sekä benchmarkingin avulla tutustuttiin Evo-metodiin (Malatouxin, 2003). Tämä yhdistelmä toi paljon teoreettista sekä käytännönläheistä tietoa Evo-metodin käytöstä sulautetun ohjelmiston suunnittelussa ja sillä olikin paljon vaikutusta syntyneeseen prosessiin. Evo-metodista saatiin alkuperäinen ajatus sykleihin ja siitä otettiin mukaan myös evoluutiomaisesti kasvava suunnittelu. Tästä on se hyöty, että kaikkea dokumentointia tai esisuunnittelua ei tarvitse tehdä etukäteen, jolloin työkuormaa voidaan tasata. Lisäksi evoluutiomainen lähestymistapa antaa mahdollisuuden aloittaa ohjelmointi, vaikka kaikki ohjelmiston ominaisuudet eivät olisikaan tiedossakaan. Tämä on myös uutta Vaisala Instruments -osastolla. Ennen prosessin mukaan määritettiin aluksi kaikki ohjelmiston ominaisuudet, mutta käytännössä muutoksia tehtiin ihan loppumetreilläkin, jotka olivat vastoin tuoteprosessia. Suunniteltu ohjelmiston suunnitteluprosessi sallii muutosten tekemisen tiettyyn mahdollisimman myöhäisessä vaiheessa olevaan katselmukseen saakka suunnitteluaikataulua vaarantamatta. Näin saadaan parempaa ohjelmiston suunnittelun hallittavuutta ja laatua, koska nyt esimerkiksi kaikkien ominaisuuksien testaamiselle jää aikaa. Ollaan siis päästy lähemmäksi ketterämpiä agile-metodeja.

XP-metodistakin otettiin mukaan muutamia hyviä ideoita. Yksi idea oli painottaa viimeistään sovellusohjelmointivaiheessa valitsemaan ne ominaisuudet, jotka pakottavat suunnittelemaan ohjelmiston arkkitehtuurin valmiiksi tässä vaiheessa. Toinen ja ehkä merkittävämpi idea oli painottaa lähdekoodin kommentoimisen tärkeyttä, koska Beckin mukaan loppujen lopuksi tärkein dokumentti on lähdekoodi. XP-metodissa tämä tapahtuu pariohjelmoinnin avulla. (Beck, 2000.) Meidän tapauksessa pariohjelmointi ei ole aina mahdollista, joten päätimme kehittää laadullisen mittarin (kts. kappaleet 6.3 ja 7.2.7), jolla voimme seurata sekä ohjata koodin kommentoinnin tasoa.

Suunniteltu prosessimittari

Mittarin kehittäminen oli erittäin haastavaa. Ongelmana oli se, että kuinka erotamme ohjelmiston osuuden tuoteprosessin lopputuotteesta. Tämä myös karsi useita kirjallisuudessa esitettyjä valmiita mittareita pois. Kirjallisuudessa mainittiin myös, että sulautetulle ohjelmistolle on vaikeaa tai jopa mahdotonta tehdä luotettavaa mittaria (Jones, 1994). Mielestämme onnistuimme kuitenkin kehittämään mittarin, joka tukee ohjelmistosuunnittelun aliprosessin laadun seuraamista. Lisäksi sitä on helppo käyttää ja se mittaa juuri oikeaa asiaa.

Laatu

Sulautetun ohjelmiston laatu- sekä turvallisuusstandardit asettivat omat vaatimuksensa prosessille. Prosessi suunniteltiinkin siten, että se täyttää ISO9001 laatustandardin ja tukee laatujohtamista. Prosessi pyrittiin myös suunnittelemaan niin, että se on tarvittaessa tulevaisuudessa muunneltavissa turvallisuusstandardit täyttäväksi. Tämä vaatimus rajasi useita agile-metodeja pois, koska esimerkiksi Boehm (2002) on todennut, että harva agile-metodi täyttää turvallisuusstandardit (Boehm, 2002). Stan-

dardit eivät toisaalta vaadi minkään määrätyn mallin kuten vesiputousmallin käyttöä, mutta niiden kriteerien täyttämisessä vesiputousmalli on selvästi lähinnä.

Kuinka prosessijohtaminen huomioitiin?

Vertaillaan seuraavaksi prosessia johtamisen kannalta. Prosessia suunniteltaessa on pyritty ottamaan huomioon uusimpia prosessiajattelulle tyyppisiä ominaisuuksia. Näitä olivat esimerkiksi valtuuttaminen, sisäisen yhteistyön parantaminen, asiakkaan huomioonottaminen, oppiminen ja johtajan uudentyyppinen rooli.

Prosessi pyrittiin suunnittelemaan siten, että sen avulla ohjelmistosuunnittelija tietää joka hetki mitkä ovat hänen tehtävänsä sekä mitä hän voi päättää. Tätä käytäntöä vahvistettiin kirjaamalla prosessimanuaaleihin kunkin prosessin aikana tehtävien toimenpiteiden tekijät.

Prosessijohtamisen tarkoituksena on parantaa organisaation sisäistä yhteistyötä (Hammer & Stanton, 1999; Hannus, 1994; Koivula & Teikari, 1996). Tähän pyrittiin hyvällä dokumentoinnilla ja erityisesti mahdollisimman joustavilla ja tehokkailla SWDR-katselmuksilla. Näiden katselmuksien byrokratia tehtiin myös mahdollisimman vähäiseksi, jotta niissä voidaan keskittyä olennaiseen eli ohjelmistosuunnittelun ohjaamiseen.

Dokumentointi ja erityisesti dokumenttipohjat ovat varmasti yksi tärkeimmistä asiantuntijaprosessin johtamisen työvälineistä. Dokumenttien avulla prosessin vastuuhenkilö voi seurata mitä asiantuntija on tehnyt ja mikä on ollut tulos. Tämän takia dokumenttipohjien tekoon sekä hiomiseen kiinnitettiin erityistä huomiota, jotta niitä käytettäessä saataisiin mahdollisimman hyviä dokumentteja. Selkeä dokumentointi toimii varmasti myös tehokkaana organisaation yhteistyötä lisäävänä tekijänä. Lisäksi hyvällä dokumentoinnilla pystytään lisäämään projekteista oppimista, joka on yksi prosessiajattelun kulmakivistä (Koivula & Teikari, 1996; Laamanen, 2001).

Merkittävä parannus ohjelmistoprosessin johdettavuudelle on asiakastarpeiden mahdollisimman tehokas huomioonottaminen (Beck, 2000; Highsmith, 1999). Vaisala Instrumentsin tapauksessa ei puhuta suorista asiakaskontakteista vaan suunnittelijan lähin asiakas on projektipäällikkö sekä tuotelinjapäällikkö. Tämä on toisaalta johtanut helpommin tilanteisiin, jossa ohjelmistomuutoksia on tehty milloin vain. Suunniteltu prosessi asettaakin selkeämmät pelisäännöt ja pyrkii näissä puitteissa sallimaan muutokset mahdollisimman pitkään.

Johtajan uusi rooli ohjelmiston suunnitteluprosesseja johdettaessa oli agile-metodien yksi eniten painottama asia (Abrahamsson et al, 2002; Beck, 2000; Highsmith, 1999). Agile-prosessia johdettaessa johtajan tulisi niiden mukaan olla enemmän valmentajatyyppinen. Tämä oli mielessäni koko prosessin suunnittelun ajan, mutta sen painottaminen itse prosessikuvauksessa tai manuaalissa on vaikeaa ja lähes mahdotonta. Jotta johtajat saavuttaisivat tarvittavat taidot ja asennemuutoksen tarvittaisiin mielestäni uusia kehityshankkeita ja koulutusta.

Prosessi yleensä

Mutta onko suunniteltu prosessi sopiva asiantuntijaorganisaatioon, eli antaako se tarpeeksi vapauksia asiantuntijoille toteuttaa tehokkaasti työtään? (Martinsuo, 2001;

Laamanen, 2001.) Mielestäni suunniteltu prosessi sopii hyvin asiantuntijaorganisaation käyttöön, koska se antaa asiantuntijalle tarpeeksi vapauksia työskennellä. Se myös tarjoaa tärkeimmät työkalut eli dokumentit, jotka on pyritty suunnittelemaan työtä mahdollisimman paljon helpottavaksi.

Kokonaisuudessaan prosessista voisi sanoa sen, että siitä tuli erittäin Vaisala Instruments -divisioonan tarpeisiin räätälöity. Kuvattu prosessi on aivan hahmottelemani kuvan 8 pyramidin alalaidassa. Tämä oli myös odotettavaa, koska prosessia rajoittivat niin tuoteprosessit kuin suhteellisen yksilölliset ja ainutlaatuiset tarpeetkin.

Prosessista löytyy kuitenkin yhtäläisyyksiä alan kirjallisuuteen, kuten tässä kappaleessa on kerrottu. Siinä on myös varmasti paljon uusia ideoita muihinkin sulautetun ohjelmiston suunnitteluprosesseihin, kuten esimerkiksi SWDR1½-katselmoinnin synkronointi elektroniikan prototyypin kanssa.

Uskon, että kuvattu prosessi on auttanut ja tulee auttamaan Vaisalan ohjelmistosuunnittelijoita hallitsemaan ja hahmottamaan omaa työtään. On kuitenkin muistettava, että prosessikuvaus on aina vain jonkin henkilön tai ryhmän tekemä hahmotelma työn kulusta, joten se tuskin koskaan pystyy täydellisesti kuvaamaan todellista tilannetta. Lopuksi voisi todeta, että hankkeessa on löydetty menetelmä (prosessi) ongelmaan, joka on vain yksi ratkaisu monien oikeiden menetelmien joukossa.

9 LOPUKSI

Pohditaan lopuksi, mitä työstä opittiin ja mitä uutta se antoi. Tässä luvussa esitetään myös vastauksia seuraaviin kysymyksiin: Mitä olisi voinut tehdä toisin? Mitä asioita jäi avoimeksi? Mitkä ovat tulevaisuuden suunnitelmat?

9.1 Mietteitä...

...Kehityshankkeesta

Tässä työssä ja kehityshankkeessa on pyritty noudattamaan kokonaisvaltaisen kehittämisen oppeja. Sen perimmäinen ajatus on ottaa huomioon niin ihmiset kuin tekniikkakin, joita tässä tapauksessa olivat ohjelmistosuunnittelijat ja prosessiteoriat (Worren et al, 1999). Tämän tyylinen myös ihmisiä huomioiva lähestymistapa oli työn alussa monelle kehitysryhmän jäsenelle uutta. Kaikesta huolimatta hankeen edetessä ja käytännön toimia tehtäessä ryhmä omaksui kokonaisvaltaisen kehittämisen tapoja. Ryhmä ei enää esimerkiksi vierastanut tiedotustilaisuuksien järjestämistä tai muita esiin nostamiani kokonaisvaltaisen kehittämisen työkaluja. Kehittämistä edesauttoi varmasti myös ryhmän sisäinen tasapaino tekniikan sekä OD:n välillä. Ryhmän jäsenillä oli myös hankkeen aikana selkeät roolit, jotka loppua kohden vahvistuivat ja näyttivät toimivan hyvin. Tässä hankkeessa mukana olo paransi mielestäni myös huomattavasti ryhmän jäsenten valmiuksia tulevaisuuden kehitystyöhön.

Suuri kysymys, jota itselleni esitin koko hankkeen ajan oli se, onko osallistaminen tarpeeksi laajaa. Lopullista vastausta tähän ei tämän diplomityön puitteissa ehditä saamaan, mutta sen voi sanoa, että osallistaminen on ollut laajempaa kuin koskaan ennen tämän divisioonan kehityshankkeissa. Divisioonalla on toisaalta pitkät perinteet kehittämisessä ja tavassa hoitaa sitä. Divisioonan kehityspäällikönkin mukaan kehityshankkeet onkin divisioonan sisällä yleisesti totuttu näkemään positiivisessa ja myönteisessä valossa sekä asiantuntijamaisesti tehtynä (Pahkala, 2003). Itse toivon, että tässä hankkeessa harjoitettu osallistaminen auttaisi jalkauttamisvaiheessa ja parantaisi esimerkiksi divisioonaan integroitujen uusien suunnittelijoiden sopeutumista.

Vaikka yleisesti ottaen hankkeeseen suhtauduttiin positiivisesti, oli myös selvästi havaittavissa todella heikkoa kiinnostusta sitä kohtaan. Tämä ilmeni esimerkiksi siten, että jopa kehitysryhmän sisällä kehitystyötä ei mielletty "oikeaksi työksi". Lisäksi ohjelmistosuunnittelijoiden kuukausikokoontumisessa ei jaksettu oikein kuunnella, saati sitten kommentoida prosessiin liittyneitä tiedotuksia. Kehitysryhmää ja ohjelmoijia tuntui motivoivan eniten erikseen prosessikehitykselle pyhitetyt tilaisuudet. Tämän takia niitä pyrittiin järjestämään ja hyödyntämään mahdollisimman paljon. Uskon itse, että niillä oli myös selvää markkinointiarvoa.

Jälkikäteen ajateltuna vision suunnitteluun ja ennen kaikkea tiedostamiseen, jota esimerkiksi Kotter pitää tärkeänä, olisi ehkä kannattanut kiinnittää enemmän huomiota (Kotter, 1995). Nykyisellä visiolla mielestäni pärjättiin ja se johti hanketta oikeaan suuntaan, tosin sen edetessä pieniä tulkintaerimielisyyksiä kyllä ilmeni. Vaisalan kehitysprosessikaan ei painota selkeästi vision tarkoitusta tai sen kirkastamisen

tärkeyttä. Nyt jos tekisin samat asiat uudestaan, olisin ehdottomasti kiinnittänyt huomiota ennen DVR1-tilaisuutta vision päivittämiseen ja kirkastamiseen.

Hanke on ainakin minulle osoittanut todeksi Kososen et al. esittämän väittämän, että yksinkertaiset kehityshankkeen mallit ovat naiiveja (Kosonen et al, 1998). Mielestäni naiivienkin mallien kattavuutta ja todenmukaisuutta voidaan jo huimasti parantaa esimerkiksi Millerin ja Greenwoodin taulukossa 1 tiivistetyillä ajatuksilla (Miller & Greenwood, 1997). Toisaalta ilman Vaisalan kuvan 2 mallia ei kehityshanketta voisi esittää kovinkaan hahmoteltavana kokonaisuutena. Paras neuvo ja oppi, jonka voisin tästä työstä antaa mallia käyttävälle tuotekehitysinsinöörille on se, että prosessin mukaan tehtyä hanketta ei voi viedä läpi kuin tuoteprojektia, vaan siinä on huomioitava myös kehityshankkeiden kompleksisuus ja ihmisten epärationaalisuus.

Edelliseen ajatukseen liittyen kehityshankkeen ja tuoteprojektin luonteiden suurin eroavaisuus tuotekehityksen kannalta on mielestäni asiakas ja sen merkityksen tiedostaminen. Tuoteprojektin tärkein asiakkaalle näkyvä tulos on tuote. Vastaavasti prosessin kehityshankkeen asiakkaalle näkyvin tuotos on prosessimanuaali. Tuoteprojekti siirtyy päättymisen jälkeen ylläpitotilaan. Samoin kehityshankeen tulisi vastaavasti siirtyä jatkuvan parantamisen tilaan. Tuoteprojektin markkinoinnin hoitaa markkinointiosasto asiakkaalle päin, mutta kehityshankkeessa markkinoijat ovatkin kehittäjiä. Tämä markkinointi eli tiedottaminen, osallistaminen, kouluttaminen yms. organisaation kehittämistoimenpiteet ovat juuri niitä asioita jotka helposti unohdetaan insinöörien kanssa toimiessa.

Kehityshankkeen mielenkiintoisimmaksi sekä opettavaisimmaksi asiaksi jäi benchmarking. Se tarjosi erittäin paljon uusia ideoita sekä oli todella motivoiva. Uskon, että sillä oli ja on todella positiivisia vaikutuksia hankkeelle. Benchmarkingista opittu toimintatapa myös varmasti auttaa uusissa vastaavanlaisissa kehityshankkeissa.

Tämä hanke oli itselleni erittäin opettava. Sen avulla pääsin kokeilemaan laajasti koulussa oppimiani organisaation kehittämisen taitoja. Se opetti minulle esimerkiksi, kuinka vaikeaa on noudattaa ihmisten kanssakäymistä opettavien kirjojen oppeja. Teknillisistä asioista on vielä suhteellisen helppo väitellä esimerkiksi kirjasta opittuja ismejä ja oppeja sanelemalla. Tällöin kyllä selviää kuka on guru. Ihmisiä taas ei ismejä latelemalla voi ohjata, niinhän kaupunkilaisjärkikin sanoo. Mielestäni on hyvä tietää perusperiaatteita ihmisten käyttäytymisestä sekä johtamisesta, mutta loppujen lopuksi vaikuttamisen taito on kuitenkin kiinni kyseisen yksilön kyvyistä, taidoista ja luonteenpiirteistä.

...Prosessista

Suunnitellussa prosessissa on ratkaistu useita ongelmakohtia. Näitä olivat esimerkiksi agile-metodien käyttö sulautetun ohjelmiston suunnittelussa sekä ketterämmän metodin sovittaminen kankeaan tuoteprosessin. Prosessi on myös suunniteltu edistämään ohjelmisto- ja elektroniikkasuunnittelijan sekä projektipäällikön välistä tiedonkulkua.

Hanke opetti ainakin minulle, että prosessin tärkeimmät komponentit ovat ehdottomasti dokumenttipohjat. Dokumentointi jo itsessään on erittäin tärkeätä, koska loppujen lopuksi prosessi tähtää ylläpidon helpottamiseen, joka ei toistettavasti onnistu

ilman hyviä dokumenttipohjia. Hyvät dokumenttipohjat ovat myös mielestäni paras tapa saada asiantuntijat noudattamaan kuvattua prosessia. Tätä on vielä korostettu sillä, että prosessin kaikki katselmukset on suunnattu lähinnä dokumenttien sisällön tarkistusta varten.

Itse prosessikuvasta tuli erittäin perinteinen ja lähes vesiputoustyyppinen, mutta uutta esimerkiksi Vaisalan tapauksessa siinä oli siihen kuvattu sykli sekä vaatimusten myöhäisempi jäädyttäminen. Sykli kuvaa mielestäni erittäin hyvin tehtäviä vaiheita ja sen avulla voidaan helposti ja nopeasti kertoa prosessin luonne sekä työn kulku. Sykliajatusta voisi mielestäni hyvin vielä jalostaa ja käyttää myös muidenkin aliprosessien työvaiheiden kuvaamiseen.

Prosessin suunnittelu sekä prosessikuvaus on varmasti auttanut muita organisaation jäseniä hahmottamaan ohjelmiston suunnittelun saloja. Mutta kokevatko ohjelmistosuunnittelijat tämän uhkaksi, koska se tekee oman ohjelmistosuunnittelun mystifioinnin vaikeammaksi? Onko tällä merkitystä? Se jää nähtäväksi.

9.2 Jatkotoimenpiteet

Tässä työssä on dokumentoitu kehityshankkeen neljä ensimmäistä vaihetta kuudesta. Tämä ei kuitenkaan tarkoita sitä, että suurin osa työstä olisi tehty. Itse asiassa erittäin suuri ja ehkä raskain osuus, nimittäin prosessin jalkauttaminen, on vielä edessäpäin (Kosonen et al, 1998). Lisäksi on muistettava mitata prosessista saatuja hyötyjä ja haittoja sekä päättää aloitettu hanke. Esittelen seuraavaksi toimenpide-ehdotuksia, joita voitaisiin käyttää esimerkiksi juuri tämän hankkeen jalkauttamisen sekä mittaroinnin toteuttamiseksi.

Jalkauttaminen

Tämän prosessin jalkauttamisvaihe on alkanut jo kauan ennen virallisen jalkauttamisvaiheen alkua esimerkiksi dokumenttipohjien julkistamisella ja käytön kokeilulla. Vaisalan toimintatavan mukaan virallinen jalkauttamisvaihe aloitetaan todennäköisesti prosessin koulutustilaisuudella. Koulutustilaisuudessa esitellään prosessi ja mittarit sekä selvitetään prosessin tarkoitusta.

Koulutus ei kuitenkaan riitä vaan muutoksen toteuttamiseksi tarvitaan vielä muutoksia toimintatavoissa sekä asenteissa (Lanning et al, 1999). Tämän toteuttamiseksi Kotterin oppeja seuraten hankkeessa tullaan kiinnittämään huomiota palkitsemiseen, kannustamiseen sekä muutoksen tiellä olevien esteiden poistamiseen. Jotta Kotterin mallit toimisivat, on muistettava vielä hänen yksi kriittisistä tekijöistä eli muutoksen taakse pitää saada vaikutusvaltaisten yrityksen avainhenkilöiden tuki. (Kotter, 1995.) Tämä tarkoittaa käytännössä sitä, että kehitysryhmän on pystyttävä ennen kaikkea myymään ajatuksensa divisioonan tuotekehityksen johdolle.

Koulutus sekä lyhyen ajan aktiivinen prosessin tukeminen ei välttämättä riitä, vaan prosessi on pystyttävä juurruttamaan yrityksen kulttuuriin (Kotter, 1995). Tämä ei kuitenkaan tule olemaan mikään helppo tehtävä (Schein, 2001). Pitkäjänteistä juurruttamista tullaan tekemään esimerkiksi mittareiden seuraamisella sekä aktiivisella prosessin päivittämisellä.

Jalkauttamisvaiheeseen kuuluu myös oleellisena osana vanhan tavan poisoppiminen (Schein, 1987). Tässä kehityshankkeen tapauksessa poisoppimista vaikeuttaa selkeästi se, ettei mitään entistä ja yhteistä käytäntöä edes ollutkaan. Suunnittelijoiden onkin opittava pois omia ja joskus omalaatuisiakin käytäntöjä. Tähän mielestäni tehokkain ase tulee olemaan uuden toimintatavan näkyvä tukeminen johtamisen sekä palkitsemisen avulla.

Kehityshankkeen mittarointi

Jotta kehityshankkeen hyödyt saataisiin kartoitettua on sen tuloksia syytä mittaroida. Mittaroinnissa on syytä muistaa tärkein asia, eli on mitattava lähtötilaa vasten eikä ideaalitilannetta (Kosonen et al, 1998). Mittaroinnilla ja tulosten (pikaisella) julkistamisella on myös tärkeä rooli prosessin jalkauttamisen onnistumisessa (Kotter, 1995).

Kehityshankkeelle ei ollut mahdollista rakentaa taloudellisia hyötyjä mittaavaa mittaria, koska sellaista ei pystytty tekemään edes suunnitteluprosessille. Tämä puolestaan johtui hybridimäisestä tuotteesta. Toisaalta ei-taloudellista toimintaa mittaavaa mittaria, kuten katselmuksien määrää mittaavaa mittaria, ei ajateltu kehittää. Tämä siksi, että suorituskyvyn ei voi ajatella automaattisesti parantuvan ajanmittaan vain tekemällä paljon ainakin periaatteessa oikeita asioita ja toimenpiteitä (Kosonen et al, 1998).

Kehityshanketta tullaankin todennäköisesti mittaamaan pehmeällä mittarilla. Tämä on ajateltu alustavasti toteuttaa yhden vuoden aikana tapahtuvilla muutamilla prosessin kommentointitilaisuuksilla. Näissä tilaisuuksissa käydään läpi prosessissa ilmenneitä epäkohtia sekä pohditaan yhdessä niihin ratkaisu. Tilaisuuden tarkoituksena on myös toimia jatkuvan parantamisen apuvälineenä.

Lopuksi voisi todeta, että vaikka kehitettäisiin minkälainen mittari tahansa, voidaan tuloksia prosessin hyödyistä tai haitoista nähdä vasta vuosien päästä. Toisaalta, tuskin koskaan voidaan määritellä juuri tämän prosessin tehokkuutta, koska vuosien päästä tehokkuuteen vaikuttavia muuttujia on valtava määrä. Tämän takia mielestäni olisi tärkeää tässä tapauksessa keskittyä välittömän palautteen keräämiseen eli nopeaan mittarointiin ja prosessin hyödyllisyyden arviointiin sitä kautta.

9.3 Kohti jatkuvaa parantamista

Tämän kehityshankkeen kohdalla voidaan mielestäni puhua radikaalista kehittämisestä. Kehityshankkeessa on kyseenalaistettu vanhoja käytäntöjä ja etsitty tilalle uusia parempia ratkaisuja, jotka ainakin Kososen et al. mukaan ovat radikaalin muutoksen tunnusmerkkejä. Tämän takia siirtyminen jatkuvan parantamisen vaiheeseen ei ole kovinkaan yksioikoista. Siihen pääsemiseksi tarvitaan mm. palkitsemisjärjestelmää, koulutusta ja pitkällisten kehittämisrutiinien hiomista. (Kosonen et al, 1998.)

Henkilöstön aktiivinen toiminta on jatkuvan parantamisen onnistumisen tunnuspiirteistä. Henkilöstön on samalla omattava ymmärrystä, halua sekä kykyä kehittää organisaatiota. (Lanning et al, 1999.) Jatkuva parantaminen on myös kytköksissä organisaation kulttuuriin ja sitä kautta muutoksen vakiinnuttamiseen. Tämä näkyy esi-

merkiksi siten, että jos uutta tehtyä toimintatapaa ei päivitetä jatkuvasti, saattaa organisaation vanha kulttuuri alkaa nostaa päätään (Kosonen et al, 1998).

Riippuen valitusta kehittämisen teorian näkökulmasta, on ideaalitapauksessa jatkuva parantaminen joko ikuista tai sitten se poikii myöhemmin uusia radikaalisia kehityshankkeita (Inns, 1996; Weick & Quinn, 1999). Vaisalan tapauksessa tavoitteena olisi päivittää prosessia vähintään kerran vuodessa eli pyrkimys olisi selvästi prosessiorientoituneeseen jatkuvaan parantamiseen.

9.4 Tulevaisuus

Mitään radikaalimpaa muutoshanketta ohjelmiston kehityksen osalta ei ole näkyvissä. Prosessia saatetaan kuitenkin muuttaa tulevaisuudessa pienillä lisäyksillä paremmin turvallisuusstandardeja ja esitutkimusta tukevaksi.

Kehitysryhmä tulee projektin päättämisen jälkeen siirtymään suurimmaksi osaksi takaisin entisiin tehtäviinsä. Kehitysryhmän jäsenet siis pääsevät kokeilemaan käytännössä oman kehitystyönsä tuloksia.

Loppusanat

Tässä työssä on esitetty Vaisala Instrumentsin -tuotekehityksen osalta paljon uusia ratkaisuja niin organisaation kehittämiseen kuin myös ohjelmiston suunnitteluprosesseihin. Toivon, että tämän hankkeen dokumentointi auttaa tulevaisuudessa myös muita vastaavanlaista kehitystyötä tekeviä.

LÄHTEET

Abrahamsson, P., Salo, O., Ronkainen, J. & Warsta, J. 2002: Agile software development methods - Review and analysis. Espoo. VTT publications 478.

Abrahamsson, Pekka 2002, VTT Elektroniikan seminaari Wanhan Sataman G-salissa 16.5.2002. Helsinki.

Amburgey, T. L., Kelly, D., & Barnett, W. P. 1993: Resetting the clock: The dynamics of organizational change and failure. Administrative Science Quarterly, 28(2), pp. 51-73.

Barret, F.J., Thomas, G.F. & Hocevar, S.P. 1995: The Central Role of Discourse in Large-Scale Change: A Social Construction Perspective. Journal of Applied Behavioral Science. Vol. 31, No. 3, pp. 352-372.

Beer, M., Eisenstat, R.A. & Spector, B. 1990: The Critical Path to Corporate Renewal. Harward Business School Press.

Beck, Kent 2000: Extreme Programming explained - Embrace Change. Reading Mass., Addisson-Wesley.

Boehm, B. W. 1988: A spiral model of software development and enhancement Computer. Vol 21., Iss. 5. pp. 61 –72.

Boehm, Barry. 2002: Get Ready for Agile Methods, with Care. IEEE Computer. pp.64-69.

Bootstrap Institute. 2003: (www-dokumentti). <http://www.bootstrap-institute.com/>. Luettu 9.5.2003.

Buchanan, D. & Badham, R. 1999: Power, Politics, and Organizational Change. Winning the Turf Game. London: Sage.

Buhanist, Paul 1999: Hajautuneen johtajuuden uudet haasteet. Artikkelipaketti TU-53.140 Johtaminen organisaatiossa. TKK.

Buhanist, P., Seppänen, L., & Virtaharju, J. 2002: Johtamisnäkemys. Artikkelipaketti TU-53.140 Johtaminen organisaatiossa. TKK.

Cockburn, Alistair 2001: Agile Software Development. Addison-Wesley.

Collins, Jim 2001: Good to Great: Why Some Companies Make the Leap... and Others Don't. Chapter 2: Level 5 Leadership. HarperCollins.

Cooper, Cary L. 1998: Theories of Organizational Stress. Manchester. Oxford University Press.

Cooper, Robert G. 1999: Product Development for the Service Sector. Massachusetts. Perseus Books. Cambridge.

Cooper, Robert G. 2000: Product Leadership: Creating and Launching Superior New Products. Massachusetts. Perseus Books. Cambridge.

Daniel, G., Boyatzin, R. & McKee, A. 2001: Primal Leadership - Realizing the Power of Emotional Intelligence. Chapter: The Neuroanatomy of Leadership Boston. Harward Business School Publishing. pp. 33-52.

Davenport, Thomas H. 1997: Ten pronciples of Knowledge Management and Four Case Studies. Knowledge and Process Management. 4 (3), pp. 187-208.

De Luca, Jeff 2003: The Latest FDD Processes. (www-dokumentti). <http://www.nebulon.com/articles/index.html> Luettu 27.5.2003.

Denton, Keith 1996: Four simple rules for leading change. Empowerment in Organization. Bradford. Vol. 4, Iss. 4, pg: 5.

Dunford, R., Dunphy, D.C. & Stace, D.A. 1990: Discussion Note: Strategies for Planned Change. An Exchange of Views Between Dunfoed, Dunphy and Stace. Organizational Studies. Vol. 11, No 1, pp. 131-136.

Dunphy, Dexter C. & Stace, Doug A. 1988: Transformational and Coercive Strategies for planned Organizational Change: Beyoun the O.D. Model. Organizational Studies. Vol. 9, No. 3, pp. 317-334.

EN 50271:2001: Electrical apparatus for the detection and measurement of combustible gases, toxic gases or oxygen - Requirements and tests for apparatus using software and/or digital technologies.

Fowler, Martin 2001: Is Design Dead? (www-dokumentti). <http://www.martinfowler.com/articles/designDead.html>. Luettu 28.5.2003.

French, W.L. & Bell, C.H. 1999: Organization Development: Behavioral Science Interventions for Organization Improvement. Sixth Ed. Upper Saddle River, NJ: Prentice-Hall. pp. 105-129.

Gilb, Tom 1997: Evo - The Evolutionary Project Managers Handbook. Evo Manuscript MINI Version. (www-dokumentti). <http://www.gilb.com/>. Luettu 15.3.2003.

Glass, R.L. 1999: The realities of software technology payoffs. Communications of the ACM. 42(2), pp. 74-79.

Glass, J.T., Ensing, M.I. & DeSanctis, G. 2003: Managing the ties between central R&D and business units. Research Technology Management. Jan/Feb. Vol. 46, No.1, pp. 24-31.

Gremba, J. & Myers, C. 1997: The IDEAL^{SM} Model: A Practical Guide for Improvement: Software Engineering Institute (SEI) publication, *Bridge*, issue three (www-dokumentti). <http://www.sei.cmu.edu/ideal/ideal.bridge.html#fmi>. Luettu 20.3.2003.

Grenning, James 2001: Launching XP at a Process-Intensive Company (www-dokumentti). <http://www.objectmentor.com/resources/articles/>. Luettu 1.6.2003.

Grenning, James 2002: Extreme Programming and Embedded Software Development. (www-dokumentti). <http://www.objectmentor.com/resources/articles/>. Luettu 25.5.2003.

Hammer, M. & Stanton, S. 1999: How process enterprises really work. Harvard business review.

Hannan, Michael T. & Freeman, John 1984: Structural Inertia and Organizational Change. American Sociological Review. Vol. 49, Iss. 2, pp 149-164.

Hannus, Jouko 1994: Prosessijohtaminen - Ydinprosessien uudistaminen ja yrityksen suorituskyky. Neljäs painos Jyväskylä. Gummerus Kirjapaino Oy.

Hansen, Morten T. 1999: The Search-Transfer Problem: The Role of Weak Ties in Sharing Knowledge Across Organizational Subunits. Administrative Science Quarttely. Vol. 44 (1), pp 82-111.

Hidding, Gezinus. J.A. 1998: Adoption of IS development methods across cultural boundaries. Communications of the ACM. Association for Information Systems. pp. 308-312.

Highsmith, James A.1999: Adaptive Software Development - A Collaborative Approach to Managing Complex Systems. New York. Dorset House Publishing.

Highsmith, J & Cockburn, A. 2001: Agile Software Development: The Businee of Innovation. Computer 34(9): pp.120-122.

Huczynski, A. & Buchanan, B. 2001: Organizational Behaviour An introduction Text, Fourth edition. London. Prentice Hall International.

Hughes, R., Ginnett, R. & Gurphy, G. 1999: Leadership is everyone's business. In Hughes, R., Ginnett, R. & Gurphy, G.: Leadership. McGraw-Hill. Boston. pp. 3-22.

IEC 61508 & BS EN 61508:2002. : Functional safety of electrical/electronic/ programmable electronic safety-related systems

IEEE Std 829-1998: IEEE Standard for Software Test Documentation

Inns, D. 1996: Organisation development as a journey. In C.Oswick and D. Grant (eds.): Organisation development. Metaphorical Explorations. pp 20-34. Pitnam: London.

ISO 9001:2000: Quality management systems - Requirements

ISO 9000-3:1997(E): Quality management and quality assuranse standards - Guidelines for the application of ISO9001:1994 to the development, supply, installation and maintenance of the computer software.

ISO/IEC 12207:1995: Information technology - Software life cycle processes. First edition.

ISO/IEC 12207:1995/Amd.1:2002(E): Information technology - Software life cycle processes. Amendment 1.

ISO 14001:1996: Environmental management systems -- Specification with guidance for use

ISO/IEC TR 15271:1998(E): Informational technology - Guide for ISO/IEC 12207 (Software Life Cycle Processes)

ISO/IEC 15504: Information Technology - Software Process Assessment

Jaaksi A., Aalto J-M., Aalto A. & Vättö K. 1999: Tried & True Object Oriented Development. Cambridge University Press.

Jeffries, R., Ann, A. & Hendrickson, C. 2001: Extreme Programming - Installed. Addison-Wesley.

Jones, Capers. 1994: Software metrics - Good, bad and missing. IEEE Computer. Vol. 27, No. 9, pp. 98-100.

Jones, Peter H. 2002: Embedded values in process and practice: Interactions between disciplinary practices and formal innovation processes. Design Management Journal. Vol. 2, pp. 20-36.

Kellner, M.I., Briand, L. & Over, J.W. 1996: A method for designing, defining, and evolving software processes Software Process,. Proceedings., Fourth International Conference, pp. 37 -48.

Koivula, Antti & Teikari, Veikko 1996: Pyramidi murenee - näkökulma tietotyön prosessijohtamiseen. Espoo. TKK, Tuotantotalous, Työpsykologia.

Komulainen, Kati 2001: Osaamisen johtaminen osana yrityksen johtamisjärjestelmää. Työn Tuuli 2 / 2001. Henry Ry.

Korhonen, Jukka & Kalaoja, Jarmo 1992: Turvallisten sulautettujen ohjelmistojen kehittäminen ja analysointi. VTT Research Notes : 1430. Espoo. VTT.

Korpi-Filppula, Kutilainen, Lanning, Rintala, Salminen & Toivanen 2000: Kehittäjän karttakirja CD-ROM - työkaluja kehitysprojektin käytännön toteutukseen. Talentum.

Kosonen, Buhanist, Kesäjärvi, Kymäläinen, Lehtonen, Salonen & Tanskanen 1998: Muutoksen etulinjassa. Hämeenlinna. Karisto.

Kotter, John P. 1991: Johtajuus menestystekijänä. Weilin+Göös. Espoo. pp. 31-51.

Kotter, John. P. 1995: Why transformation efforts fail. Harward Business Review, Mar-Apr 1995. pp. 59-67.

Krogh von Georg 1998: Care in Knowledge Creation. California Management Review. Vol. 40(3), pp 133-153.

Kärnä, p. & Aro, A. 2002: Työkontekstin ja työuupumuksen yhteys. Psykologia 4/2002.

Laamanen, Kai 2001: Johda liiketoimintaa - Prosessien verkkona. Helsinki. Laatukeskus.

Laatukeskus. 1998: Oy Benchmarking käsikirja: Nopea oppiminen - ylivoimainen kilpailuetu. Lahti. Esa Print Oy.

Lanning H., Roiha M. & Salminen A. 1999: Matkaopas muutokseen - miten kehität organisaatiota tehokkaasti ja hallitusti. Helsinki. Kauppakaari.

Lanning, Harri 2001: Planning and implementing change in organisations : a construct for managing chance projects. Espoo. Helsinki University of Technology.

Laplante, Phillip A. 1997: Real-time Systems Design and Analysis - An Engineer's Handbook New York. IEEE Press Marketing.

Lecklin, Olli 2002: Laatu yrityksen menestystekijänä. Helsinki. Kauppakaari. Talentum Media Oy.

Liberty, Jesse 1996: Opeta itsellesi C++. Jyväskylä. Gummerus Kirjapaino Oy.

Malatouxin Niels 2003,(www-dokumentti). < http://www.malotaux.nl/>. Luettu 15.5.2003.

Mann, Charles C. 2002: Onneton ohjelmointi vie rahat ja hermot. Tiede nro. 7 / 2000. Sanoma Magazines Finland Oy. pp 28-33.

Martinsuo, Miia 2001: Tuotekehitysorganisaatioiden erityispiirteet ja odotukset henkilöjohtamiselta. Työn Tuuli 2 / 2001. Henry Ry.

Martinsuo, Miia 2003: Luentomonisteet T&K-johtaminen kurssi 10.3.2003. Otaniemi.

McDermid, J.A. & Pumfrey, D.J., 2001: Software Safety: Why is there no Consensus?, Proceedings of the 19th International System Safety Conference, Huntsville, AL, System Safety Society, (www-dokumentti).
<http://www-users.cs.york.ac.uk/~djp/>. Luettu 30.5.2003.

McFeeley, Bob 1996: IDEAL: A User's Guide for Software Process Improvement CMU/SEI-96-HB-001 Pittsburgh, Pa: The Software Engineering Institute (SEI), Garnegie Mellon University.

Miller, D. & Greenwood, R. 1997: Creative Chaos Versus Munificent Momentum - The Schism Between Normative and Academic Views of Organizational Change. Journal of Management Inquiry. Vol. 6, No. 1, pp. 71-78.

Mäkinen, Terhi 2002: IT-alan nuoret uupumuksen partaalla. Työ, terveys, turvallisuus. 2/2002.

Mäkäräinen, Minna 2000: Software change management processes in the development of embedded software. VTT Publications : 416. VTT. Espoo.

Nahapiet J. & Ghoshal S. 1998: Social Capital, Intellectual Capital, and the organizational Advantage. Academy of Management Review. Vol. 40(3), pp 40-54.

Nonaka, I., Ryoko, T. & Konno, N. 2000: SECI, BA, and Leadership: a Unified Model of Dynamic Knowledge Creation. Long Range Planning. Vol. 33(1), pp 5-34.

Pahkala, Samuli 2003: Development Manager. Vaisala Oyj. Haastattelu 21.2.2003

Pulli, P., Elmstrom, R. & Gonzalo, L. 1994: IPTES - Incremental Prototyping Technology for Embedded real-time Systems. (www dokumentti). <http://citeseer.nj.nec.com/pulli91iptes.html>. NEC Research Institute CiteSeer. Luettu 24.5.2003.

Rautiainen, Kristian 2002: T- 76.631 Software Processes (2) L Introduction. Opetusmoniste. 24.10.2002. Otaniemi.

Ronkainen, J., Taramaa, J. & Savuoja, A. 2002: Characteristics of Process Improvement of Hardware-Related SW. Rovaniemi. In Profes 2002. Rovaniemi, Finland. pp.247-257.

Sample, Steven B. 2002: The Contrarian's Guide to Leadership. Chapter 6: Give the Devil His Due. Wiley Europe. pp. 91-105; 189-192.

Schein, Edgar H. 1987: Process Consultation - Lessons for Managers and Consultants vol. 2. London. Sage.

Schein, Edgar H. 2001: Yrityskulttuuri - selviytymisopas. Tampere. Suomen Laatukeskus Koulutuspalvelut Oy.

Smart, A., Maull, R., Childe, S., Bennett, J. & Weaver, A. 1996: Specification Book Three - Report on Process Analysis Techniques, Working Paper WP/GR/J95010-3.University of Plymouth.

Software Engineering Institute. 2003: (www-dokumentti). < http://www.sei.cmu.edu/sei-home.html>. Luettu 9.5.2003.

Software Process Improvement and Capability dEtermination. 2003: (www-dokumentti). <http://www.sqi.gu.edu.au/spice/>. Luettu 9.5.2003.

Sutherland, V.J., & Cooper, C.L. 2000: Strategic Stress Management - an organizational approach. London. Macmillan Press LTD.

Teikari, Veikko 1999: Asiantuntijayhteisössä johtaja luo haasteita ja jatkuvuutta. Polysteekki. 4/99.

Teikari, Veikko 2000: Dualistisen johtamisen haasteet. Hetkyn Tietosanomat 2/2000.

Teikari, Veikko 2002: Työn etiikka. Kirjassa P. Juuti (toim.) Ethosta etsimässä. Puheenvuoroja johtamisesta ja yrittämisen etiikasta. Juva. Ps-kustannus. Aavaranta-sarja. No. 50.

Teikari, Veikko 2003: Professori. Työpsykologian ja johtamisen laboratorio, Teknillinen korkeakoulu. Sähköposti. 28.2.2003.

Vaisala, 2000: Development Process - Frisk & Pankakoski. Sisäinen dokumentti. Vantaa.

Van De Ven, A., Poole, M. 1995: Explaining Development and Change in Organizations. Academy of Management Review, Vol. 20, No. 3, pp. 510-540.

Weick, K. & Quinn, R. 1999: Organizational Change And Development. Annual Review of Psychology 1999, pp. 361-386.

Worren, N., Ruddle, K. & Moore, K. 1999: From organizational development to change management: The emergence of a new profession. The Journal of Applied Behavioral Science. Vol. 35, Iss. 3, pp. 273-286.

Wil van der Aalst & Kees van Hee 2002: Workflow Management - Models Methods and Systems. London. The MIT Press.

Yukl, Gary 1998: Leadership in organizations. Chapter 13: Transformational and Cultural Leadership. New Jersey. Prentice Hall. pp. 324-346.

Yli-olli Pekka & Hokkanen Timo 1991: Softa 9000 - Ohjelmistotuotannon laadun kehittäminen. Espoo. Mecrastor Oy.

LIITTEET

Liite 1, Workshopin aikataulu ja sisältö

VAISALA

SKP

Softakehitysprosessin kehitysprojekti:

- Vuoden 2003 kehityshanke, jonka tuloksena syntyy softakehitysprosessin kuvaus
- Kuvausta täydennetään myöhemmin ilmenevien tarpeiden mukaan

2003-03-10 PTu

VAISALA

SKP

Tavoitteet:

- Kuvata ja tarvittavilta osin luoda ohjelmistokehityskäytäntömme ja -tapamme
- Ohjeet ja suositukset helpottavat yhteistyötä
- Tavoitteena ei ole muuttaa hyväksi todettavia työskentelytapoja
- Varmistaa, että projektien vetäjät ymmärtävät ohjelmistokehityksen ominaispiirteet
- Standardit ja muut vaatimukset vaativat kehityskäytäntöjen kuvauksen

2003-03-10 PTu

VAISALA

The Joel Test

1. Do you use source control?
2. Can you make a build in one step?
3. Do you make daily builds?
4. Do you have a bug database?·
5. Do you fix bugs before writing new code?
6. Do you have an up-to-date schedule?·
7. Do you have a spec?·
8. Do programmers have quiet working conditions?
9. Do you use the best tools money can buy?
10. Do you have testers?
11. Do new candidates write code during their interview?
12. Do you do hallway usability testing?·

VAISALA

SKP

Current Status Workshop:

- Yhteiset ja henkilökohtaiset softakehitystavat
- Auttaa välttämään ristiriidat nykykäytäntöjen kanssa
- Selvittää toisillemme, mitä tapoja kullakin on softakehitystyössään

VAISALA

SKP

Current Status Workshop Agenda:

12:00	Agenda and general info, SDP process presentation	15 min
12:30	Individual work	15 min
12:45	Pair working	30 min
13:15	-Break-	15 min
13:30	Group working	1 h
14:30	Presentations and discussions	1-2 h
16:00	➢ informal...	

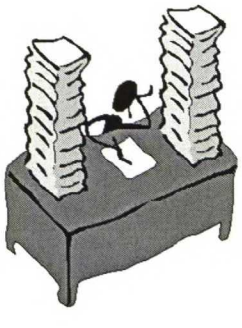
2003-03-10 PTu

VAISALA

Työskentelyohjeet

Tehtävät:

1. Täytä kysymyslomake kokemuksillasi.
   - Mitä siis teet/teit, EI miten pitäisi tehdä!



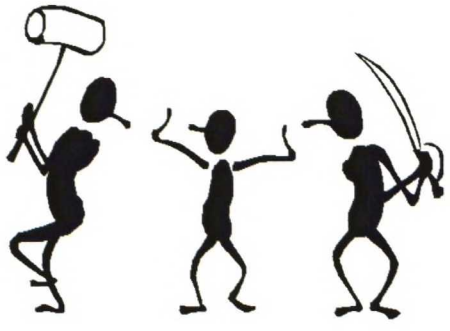
2003-03-25

VAISALA

Työskentelyohjeet

Muodostetaan parit (2 henk./pari)

2. Keskustele parisi kanssa kyselylomakkeen kysymyksistä.
   - Kirjatkaa ylös lomakkeen kääntöpuolelle yhteisiä tai erottavia asioita joita löysitte!



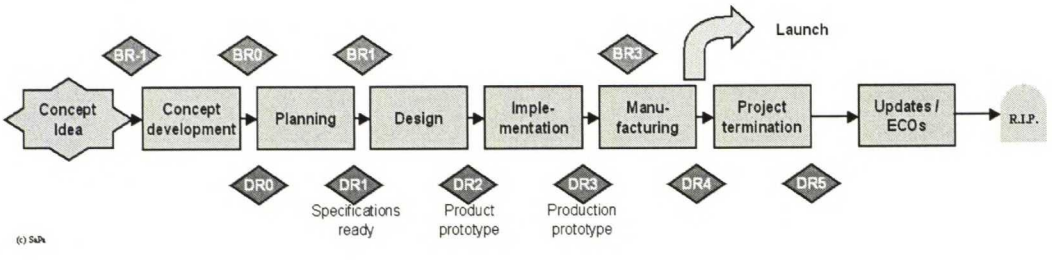
2003-03-25

VAISALA

Työskentelyohjeet

Muodostetaan ryhmät (3 - 4 henk./ryhmä)

3. Hahmotelkaa ryhmänne kanssa <u>nykyinen</u> sulautetun ohjelmiston "suunnittelun kulku".
   - Käyttäkää apunanne oheista kuvausta sekä omia että ryhmän muistiinpanoja.



2003-03-25

Liite 2, Henkilökohtaiset kysymykset

SULAUTETUN OHJELMISTON SUUNNITTELU-KÄYTÄNTÖJEN NYKYTILAN ARVIO

Kysymyksiin ei ole olemassa yhtä oikeaa vastausta. Vastaamista voi helpottaa, jos ajattelet yhtä alusta loppuun läpikäymääsi suunnitteluprosessia. Pyri vastaamaan lyhyesti sekä selkeästi!

Vastaa mitä teit/teet, EI miten pitäisi tehdä !

1. Monta sulautetun ohjelmiston suunnitteluprojektia olet Vaisalassa tehnyt _____

2. Mistä alkaa sulautetun ohjelmiston suunnittelu?

_____

3. Mistä se saa rajoitteensa/suuntaviivansa? (projektisuunnitelma, tekniset speksit,...)

_____

4. Mihin sulautetun ohjelmiston suunnittelu loppuu?

_____

5. Mikä on SW:n "status" tuoteprosessin eri DR-vaiheissa? Tätä kysymystä helpottaa, jos katsot liitettä 1.

DR0_____
DR1_____
DR2_____
DR3_____
DR4_____
DR5_____

_____

6. Mitä eri sulautetun ohjelmiston dokumentteja/asiakirjoja/muistioita/yms. suunnittelun aikana syntyy?

Dokumentin/ asiakarjan "nimi"	**Tekijä**	**Missä vaiheessa se on syntynyt?**	**Mihin se tallennetaan?**	**Miksi se on tehty ja kenelle?**

7. Missä prosessin ja suunnittelun vaiheissa testataan sulautettua ohjelmistoa ja miten?

_____

8. Miten teet versionhallintaa ja change logia? Mieti myös missä vaiheessa teet näitä.

_____

9. Mikä käynnistää ylläpitotoimet ja milloin?

_____

10. Mieti mitä eri suunnittelu- ja työvaiheita sulautetun ohjelmiston suunnitteluun kuuluu!

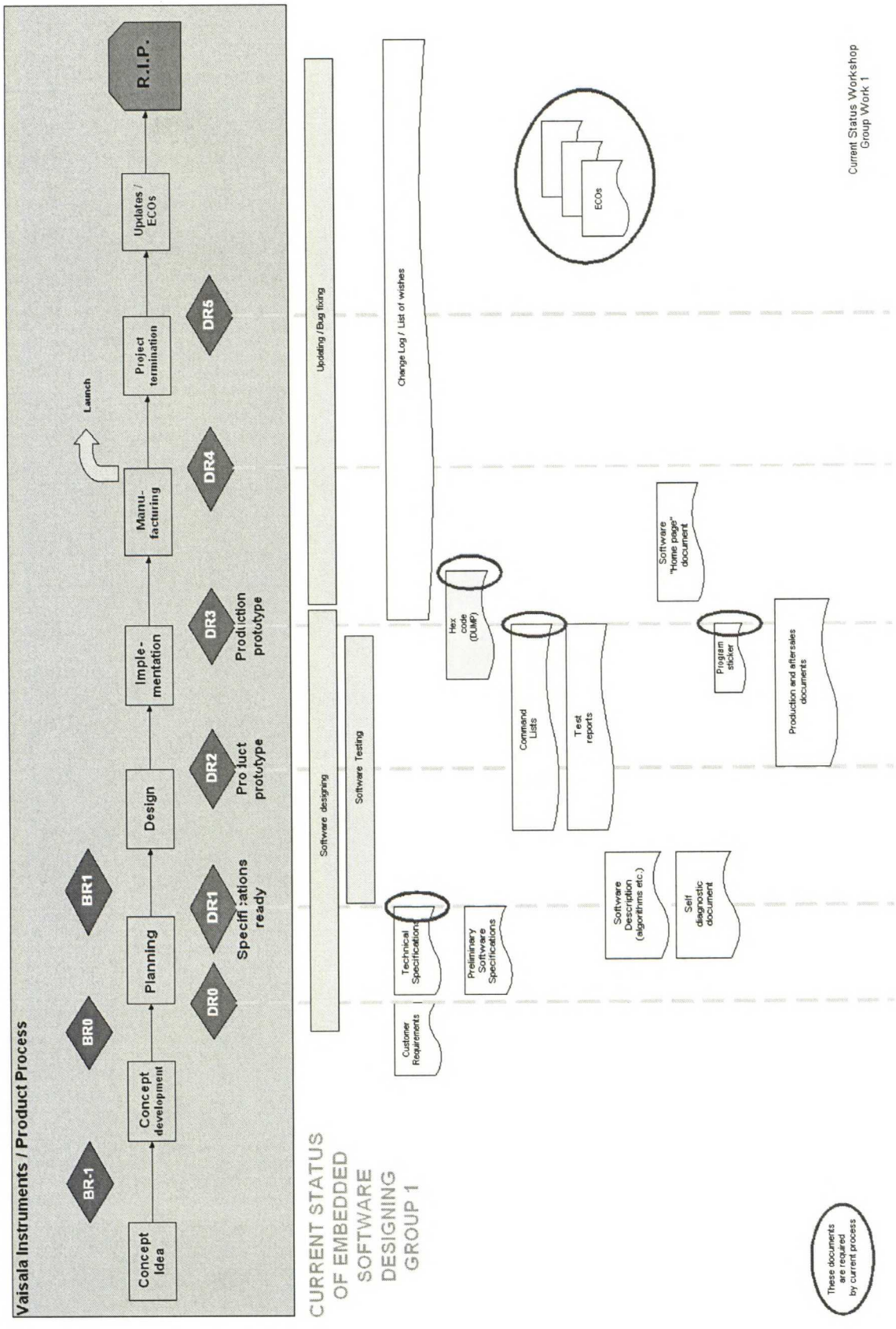
_SW speksien teko , Käyttöliittymän testaus, Change Login pito ,

_____, _____, _____,

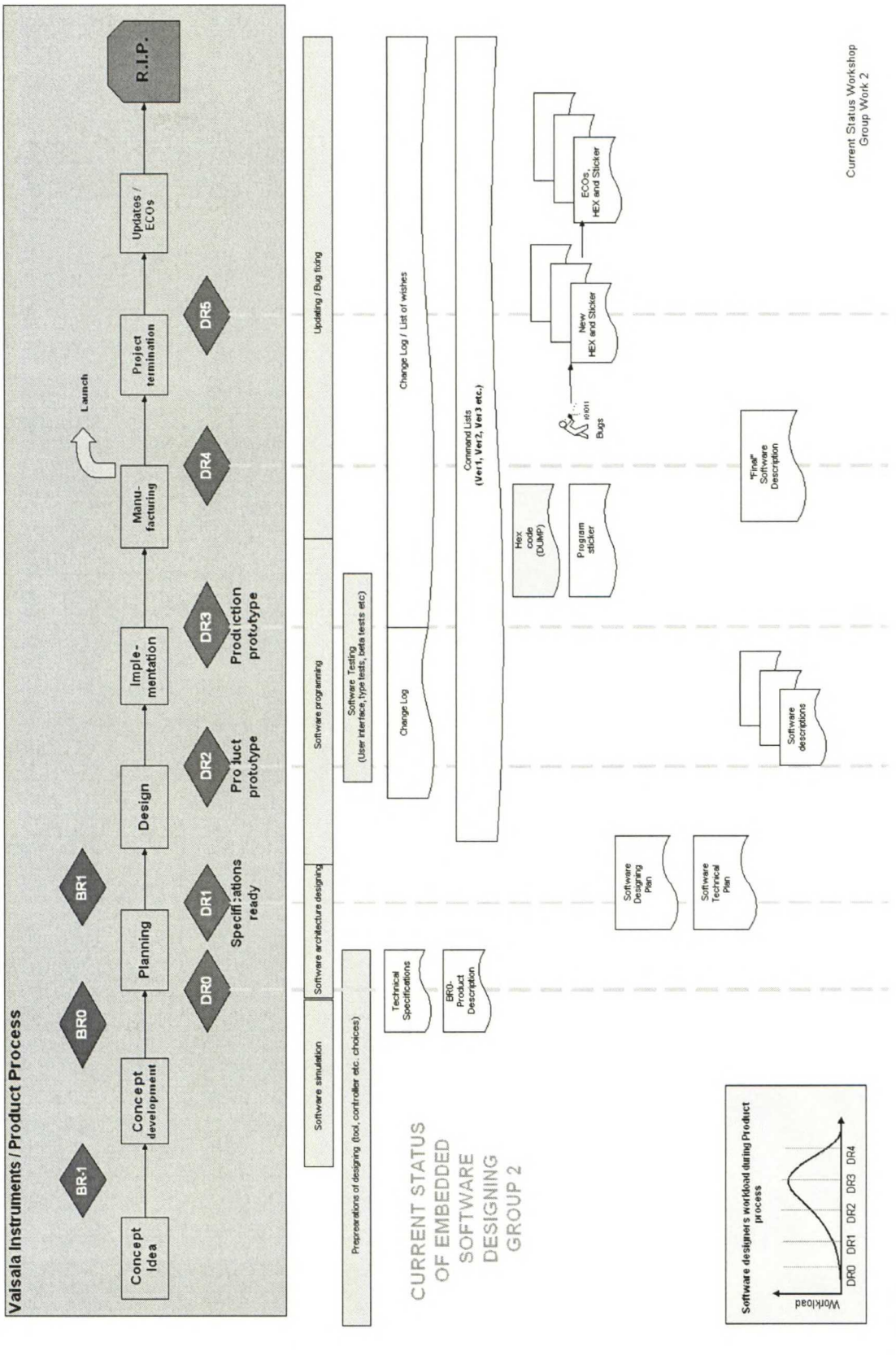
11. Kuka kysyy ohjelmistosuunnittelun tuotoksia, eli kenelle teet ohjelman?

_____

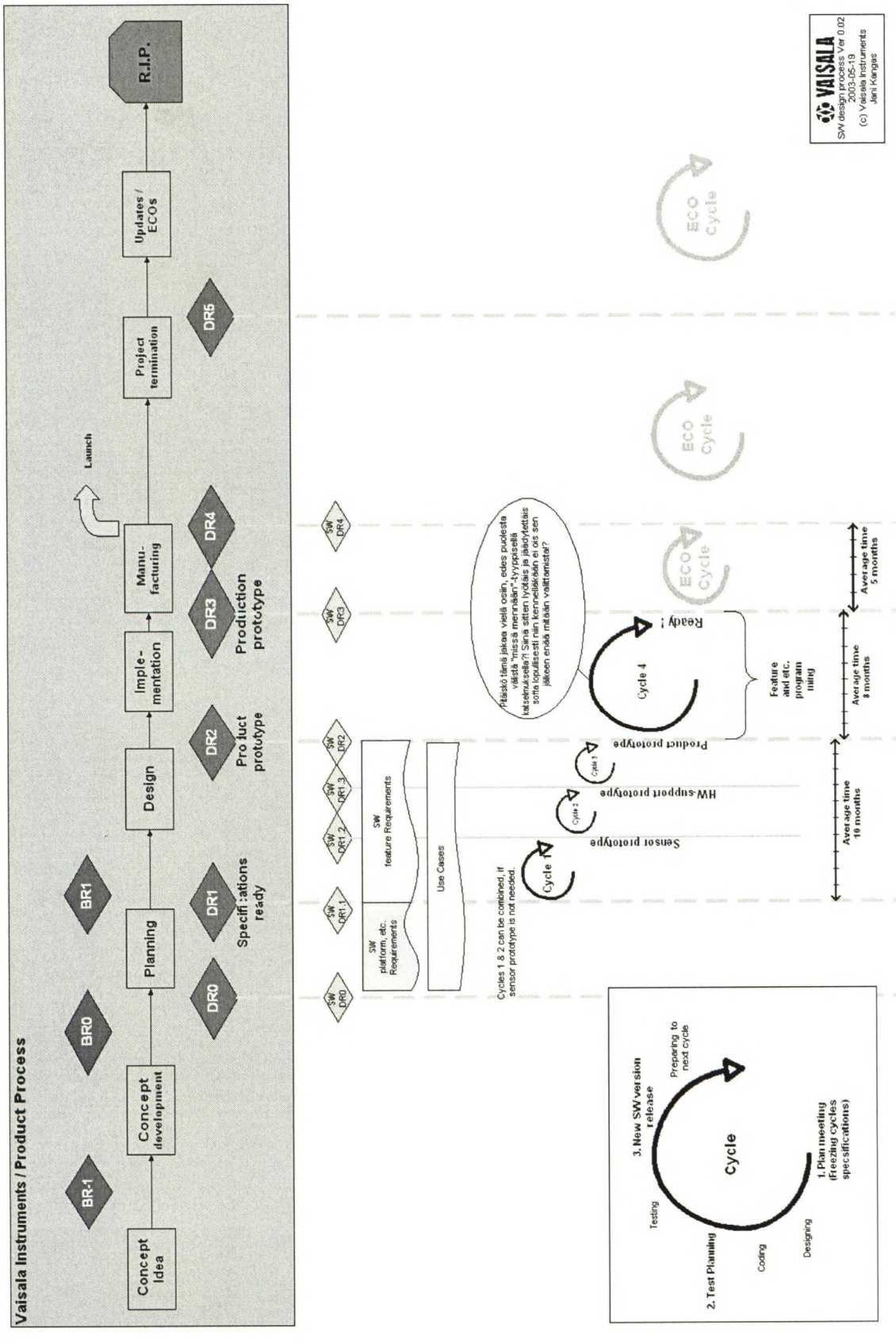
Liite 3, Ryhmän 1 tekemä nykytilan prosessikuvaus



Liite 4, Ryhmän 2 tekemä nykytilan prosessikuvaus



Liite 5, Ensimmäinen kehitysryhmän yhteinen luonnos suunnitteluprosessista



Liite 6, FAQ-kysymykset

Miksi tällainen prosessi pitää olla?
Prosessi...

- helpottaa työtä
- luo selkeät pelisäännöt
- auttaa dokumentoinnissa
- auttaa ylläpidossa
- edistää tiedonkulkua
- helpottaa copy-paste operaatioita projektien välillä
- vähentää vatsanpuruja (ISO) auditin iskiessä
- helpottaa uusien softakaverien sopeutumista

Miten prosessi vaikuttaa elämään?
Pitää...

- seurata prosessia ja ymmärtää missä vaiheessa ollaan menossa
- kirjoitella ja ylläpitää dokumentteja
- esitellä dokumentteja
- järjestää katselmointeja
- kieltäytyä muutoksista

Milloin teen dokumentteja?

- Prosessikaavio ja -manuaali auttavat ja kertovat milloin niiden pitää olla tehtynä
- Dokumentointia suunniteltaessa on myös ajateltu työn kuormitusta. Joten täyttämällä dokumentit prosessin mukaan välttyy suurelta kuormitushuipulta

Minun pitäisi tehdä dokumentti, mutta mitä siihen kirjoitan?

- Prosessi sisältää jokaiseen vaadittuun dokumenttiin pohjan. Nämä pohjat opastavat ja antavat vihjeitä kuinka ne täytetään
- Perusperiaate kuitenkin on, että dokumentteihin kannattaa kirjoitetaan vain hyödyllisiä asioita.
- Dokumentteja kirjoittaessa on tärkeää pitää mielessä niiden tehtävä eli myöhemmin tapahtuvan ylläpidon tukeminen ja helpottaminen

Minulla on täytetty dokumentti, mutta mihin laitan sen?

- Prosessimanuaalissa on ohjeet tallennuspaikoista ja siitä mitä versiota ja tiedostoja sinne laitetaan.

Kenelle dokumentit ovat tarkoitettu? Kuka niitä lukee?

- Sinä itse, kun kolmen vuoden kuluttua palaat asiaan
- Toinen suunnittelijaraukka, joka joutuu jatkamaan joskus työtäsi
- Projektipäällikkö, joka voi onnellisena esitellä täydellistä dokumentaatiota
- ISO-9000 tarkastaja, joka tulee hyvälle mielelle nähdessään prosessin ja siinä syntyvät dokumentit